aws

開發人員指南

Amazon Textract

Amazon Textract: 開發人員指南

Table of Contents

什麼是 Amazon Textract？

Amazon Textract 讓您在應用程式中新增文件文字偵測與分析變得更容易。使用 Amazon Textract 買家可以：

- 檢測各種文檔中的鍵入和手寫文本，包括財務報告、醫療記錄和稅務表格。
- 使用 Amazon Textract 文檔分析 API 從包含結構化數據的文檔中提取文本、表單和表格。
- 使用分析費用 API 處理發票和收據。
- 使用 AnalyzeID API 處理由美國政府頒發的駕駛執照和護照等身份證文件。

Amazon Textract 以 Amazon 電腦視覺科學家所開發的深入學習技術為基礎，不但經過驗證且可高度擴展，每天可分析數十億張影像。無需任何機器學習專業即可使用。Amazon Textract 包括簡單易用的 API，可以分析圖像文件和 PDF 文件。Amazon Textract 不斷從新資料中學習內容，Amazon 也會在服務中持續加入新功能。

以下是使用 Amazon Textract 的常用案例：

- 創建智能搜索索引— 使用 Amazon Textract，您可以創建圖像和 PDF 文件中檢測到的文本庫。
- 使用智能文字提取進行自然語言處理 (NLP)— Amazon Textract 讓您控制如何將文本分組為 NLP 應用程序的輸入。可以將文字和行提取文字。如果啟用了 Amazon Textract 文檔表分析，它還會按表格單元格對文本進行分組。
- 加快不同來源數據的捕獲和標準化— Amazon Textract 支持從各種文檔（如財務文檔、研究報告和醫療筆記）中提取文本和表格數據。藉助 Amazon Textract 分析文檔 API，您可以輕鬆快速地從文檔中提取非結構化和結構化數據。
- 自動從表單捕獲數據— Amazon Textract 允許從表單中提取結構化數據。藉助 Amazon Textract 分析 API，您可以將提取功能構建到現有業務工作流程中，以便通過表單提交的用户數據可以提取為可用格式。

使用 Amazon Textract 的若幹優點包括：

- 將文檔文本檢測集成到您的應用中— Amazon Textract 使用簡單的 API 讓您能輕鬆在應用程式中建置文字偵測功能，操作不再繁瑣複雜；無需具備電腦視覺或深入學習的專業知識，也能使用 Amazon Textract 來檢測文件文字。搭配 Amazon Textract 文字 API 使用，便能輕鬆在任何網路、手機或已連接裝置的應用程式中建置文字偵測功能。

- 可擴展的文件分析— Amazon Textract 使您能夠快速分析和從數百萬個文檔中提取數據，從而加快決策制定速度。
- 低成本— 使用 Amazon Textract，您僅需按您分析的文件付費。沒有最低費用或者預付款項。您可以免費試用！隨着需求成長，搭配我們的分層定價模式，即可省下更多成本。

通過同步處理，Amazon Textract 可以針對延遲至關重要的應用程序分析單頁文檔。Amazon Textract 還提供異步操作，以擴展對多頁文檔的支持。

第一次使用的 Amazon Textract 用户

如果這是您第一次使用 Amazon Textract，我們建議您依序讀以下章節：

1. Amazon Textract 工作方式— 此章節介紹 Amazon Textract 元件及其如何協同工作，實現端對端體驗。
2. Amazon Textract 入門— 此章節中，您可以設定您的賬户和測試 Amazon Textract API。

將 Amazon Textract 與AWSSDK

AWS 軟體開發套件 (SDK) 適用於許多常用的程式設計語言。每個 SDK 都提供 API、程式碼範例和說明文件，讓開發人員能夠更輕鬆地以偏好的語言建置應用程式。

SDK 文件	程式碼範例
AWS SDK for C++	AWS SDK for C++程式碼範例
AWS SDK for Go	AWS SDK for Go程式碼範例
AWS SDK for Java	AWS SDK for Java程式碼範例
AWS SDK for JavaScript	AWS SDK for JavaScript程式碼範例
AWS SDK for .NET	AWS SDK for .NET程式碼範例
AWS SDK for PHP	AWS SDK for PHP程式碼範例
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3)程式碼範例
AWS SDK for Ruby	AWS SDK for Ruby程式碼範例

可用性範例

找不到所需的內容嗎？ 請使用本頁面底部的提供意見回饋連結申請程式碼範例。

Amazon Textract 工作方式

Amazon Textract 使您能夠檢測和分析單頁或多頁輸入文檔中的文本（請參閱輸入文檔。

Amazon Textract 為以下操作提供操作。

- 僅檢測文本。如需更多詳細資訊，請參閱。偵測文字。
- 檢測和分析文本之間的關係。如需更多詳細資訊，請參閱。分析文檔。
- 檢測和分析發票和收據中的文本。如需更多詳細資訊，請參閱。分析發票和收款。
- 檢測和分析政府身份證件中的文本。如需更多詳細資訊，請參閱。分析身分文件。

Amazon Textract 提供同步操作，用於處理小型單頁文檔和接近實時響應。如需詳細資訊，請參閱 使用同步操作處理文檔。Amazon Textract 還提供異步操作，您可以使用這些操作來處理更大的多頁文檔。異步回應不是實時的。如需詳細資訊，請參閱 使用異步操作處理文檔。

當亞 Amazon Textract 操作處理文檔時，結果會在the section called “Block”物件或the section called “ExpenseDocument”物件。這兩個對象都包含檢測到的有關項目的信息，包括它們在文檔中的位置以及它們與文檔上其他項目的關係。如需詳細資訊，請參閱 Amazon Textract 應物件。如需展示如何使用Block對象，請參閱教學課程。

主題

偵測文字

Amazon Textract 提供僅返回文檔中檢測到的文本的同步和異步操作。對於這兩組操作，以下信息以多個the section called “Block”物件。

- 檢測到的文本的行和單詞

- 檢測到的文本的行和單詞之間的關係
- 檢測到的文本顯示在
- 文檔頁面上文本行和單詞的位置

如需詳細資訊，請參閱 the section called “文本的行和單詞”。

要同步檢測文本，請使用DetectDocumentTextAPI 操作，並將文檔文件作為輸入傳遞。整個結果集由操作返回。如需詳細資訊和範例，請參閱 使用同步操作處理文檔。

> **Note**
>
> 亞馬遜回應 API 運作`DetectText`不同於`DetectDocumentText`。您使用`DetectText`來檢測實時場景中的文本，例如海報或路標。

若要異步檢測文字，請使用StartDocumentTextDetection開始處理輸入文檔文件。若要獲取結果，請調用GetDocumentTextDetection。結果將在一個或多個響應中返回`GetDocumentTextDetection`。如需詳細資訊和範例，請參閱 使用異步操作處理文檔。

分析文檔

Amazon Textract 分析文檔和表單中檢測到的文本之間的關係。Amazon Textract 分析操作返回三類文檔提取 — 文本、表單和表格。發票和收據的分析通過不同的流程處理，有關詳細信息，請參閱分析發票和收款。

文字提取

從文檔中提取的原始文本。如需詳細資訊，請參閱「」文本的行和單詞。

表單提取

表單數據鏈接到從文檔中提取的文本項。Amazon Textract 將表單數據表示為金鑰/值對。在以下示例中，Amazon Textract 檢測到的文本行之一是名稱: Doe。Amazon Textract 還標識了一個密鑰（名稱:）和一個值（Doe。如需詳細資訊，請參閱「」表單數據（鍵值對）。

名稱: Doe

地址：123 任何街, 安城, 美國

出生日期：12-26-1980

鍵值對還用於表示從表單中提取的複選框或選項按鈕（單選按鈕）。

男性：☑

如需詳細資訊，請參閱「」選擇元素。

表格提取

Amazon Textract 可以提取表格、表格單元格和表格單元格中的項目，並且可以編程以返回 JSON、.csv 或 .txt 文件中的結果。

名稱	Address
安娜·卡羅萊納州	123 任何城市

如需更多詳細資訊，請參閱資料表。也可以從表格中提取選擇元素。如需詳細資訊，請參閱「」選擇元素。

對於分析商品，Amazon Textract 以多個方式返回以下內容the section called “Block”物件：

- 檢測到的文本的行和單詞
- 檢測到的項目的內容
- 檢測到的項目之間的關係
- 檢測到項目的頁面
- 項目在文檔頁面上的位置

您可以使用同步或異步操作來分析文檔中的文本。要同步分析文本，請使用AnalyzeDocument操作，並將文檔作為輸入傳遞。AnalyzeDocument返回整組結果。如需詳細資訊，請參閱 使用 Amazon Textract 分析文檔文本。

若要異步檢測文字，請使用StartDocumentAnalysis以開始處理。若要獲取結果，請調用GetDocumentAnalysis。結果將在一個或多個響應中返回GetDocumentAnalysis。如需詳細資訊和範例，請參閱 檢測或分析多頁文檔中的文本。

要指定要執行的分析類型，可以使用FeatureTypes列表輸入參數。將 TABLE 添加到列表以返回有關在輸入文檔中檢測到的表的信息，例如，表格單元格、單元格文本和單元格中的選擇元素。添加 FORM 以返回單詞關係，例如鍵值對和選擇元素。要執行這兩種類型的分析，請將表和表格添加到FeatureTypes。

在文檔中檢測到的所有行和單詞都包含在響應中（包括與FeatureTypes。

分析發票和收款

Amazon Textract 從幾乎任何發票或收據中提取相關數據，例如聯繫信息、購買的商品和供應商名稱，而無需任何模板或配置。發票和收據通常使用各種佈局，因此很難大規模手動提取數據。Amazon Textract 使用 ML 來瞭解發票和收據的上下文，並自動提取發票或收據日期、發票或收據編號、商品價格、總金額和付款條件等數據，以滿足您的業務需求。

Amazon Textract 還會識別對您的工作流程至關重要但未明確標記的供應商名稱。例如，Amazon Textract 可以在收據上找到供應商名稱，即使該名稱僅在頁面頂部的徽標中顯示，而沒有明確的鍵值對組合。Amazon Textract 還可以讓您輕鬆整合不同收據和發票的輸入，這些收據和發票為同一概念使用不同詞語。例如，Amazon Textract 會映射不同文檔中的字段名稱之間的關係，例如客戶編號、客戶編號和賬戶 ID，並將標準分類輸出為INVOICE_RECEIPT_ID。在這種情況下，Amazon Textract 會在不同文檔類型中一致地表示數據。與標準分類不一致的字段被歸類為OTHER。

以下為分析費用目前支持的標準字段清單：

- 廠商名稱：VENDOR_NAME
- 資料總數：TOTAL
- 接收方地址：RECEIVER_ADDRESS
- 發票/接收日期：INVOICE_RECEIPT_DATE
- 發票/收據編號：INVOICE_RECEIPT_ID
- 付款條件：PAYMENT_TERMS
- 小計：SUBTOTAL
- 截止日期：DUE_DATE
- 務：TAX
- 發票納稅人編號（SSN/ITIN 或 EIN）：TAX_PAYER_ID
- 項目名稱：ITEM_NAME
- 商品價格：PRICE
- 商品數量：QUANTITY

分析費用 API 返回給定文檔頁面的以下元素：

- 頁面內的收據或發票數量，表示為ExpenseIndex

- 單個字段的標準化名稱，表示為Type
- 字段的實際名稱，如文件中所顯示的欄位名稱，表示為LabelDetection
- 對應字段的值表示為ValueDetection
- 所提交文檔中的頁數表示為Pages
- 檢測到字段、值或行項目的頁碼，表示為PageNumber
- 幾何，其中包括頁面上各個字段、值或行項目的邊界框和座標位置，表示為Geometry
- 與文檔上檢測到的每個數據段相關聯的置信度分數，表示為Confidence
- 購買的單個行項目的整行，表示為EXPENSE_ROW

以下是由分析費用處理的收據的 API 輸出的一部分，顯示合計：55.64 在文檔中提取為標準字段TOTAL, 文檔上的實際文本為「總」, 置信度分為「97.1」, 頁碼「1」, 總值為「$55.64」和邊界框和多邊形座標:


```
{
    "Type": {
        "Text": "TOTAL",
        "Confidence": 99.94717407226562
    },
    "LabelDetection": {
        "Text": "Total:",
        "Geometry": {
            "BoundingBox": {
                "Width": 0.09809663146734238,
                "Height": 0.0234375,
                "Left": 0.36822840571403503,
                "Top": 0.8017578125
            },
            "Polygon": [
                {
                    "X": 0.36822840571403503,
                    "Y": 0.8017578125
                },
                {
                    "X": 0.466325044631958,
                    "Y": 0.8017578125
                },
                {
                    "X": 0.466325044631958,
                    "Y": 0.8251953125
```

```
                },
                {
                    "X": 0.36822840571403503,
                    "Y": 0.8251953125
                }
            ]
        },
        "Confidence": 97.10792541503906
},
        "ValueDetection": {
            "Text": "$55.64",
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.10395314544439316,
                    "Height": 0.0244140625,
                    "Left": 0.66837477684021,
                    "Top": 0.802734375
                },
                "Polygon": [
                    {
                        "X": 0.66837477684021,
                        "Y": 0.802734375
                    },
                    {
                        "X": 0.7723279595375061,
                        "Y": 0.802734375
                    },
                    {
                        "X": 0.7723279595375061,
                        "Y": 0.8271484375
                    },
                    {
                        "X": 0.66837477684021,
                        "Y": 0.8271484375
                    }
                ]
            },
        "Confidence": 99.85165405273438
},
"PageNumber": 1
}
```

您可以使用同步操作來分析發票或收據。若要分析這些單據，您可以使用 AnalyzeFesments 操作並將收據或發票傳遞給它。AnalyzeExpense返回整組結果。如需詳細資訊，請參閱 使用 Amazon Textract 分析發票和收據。

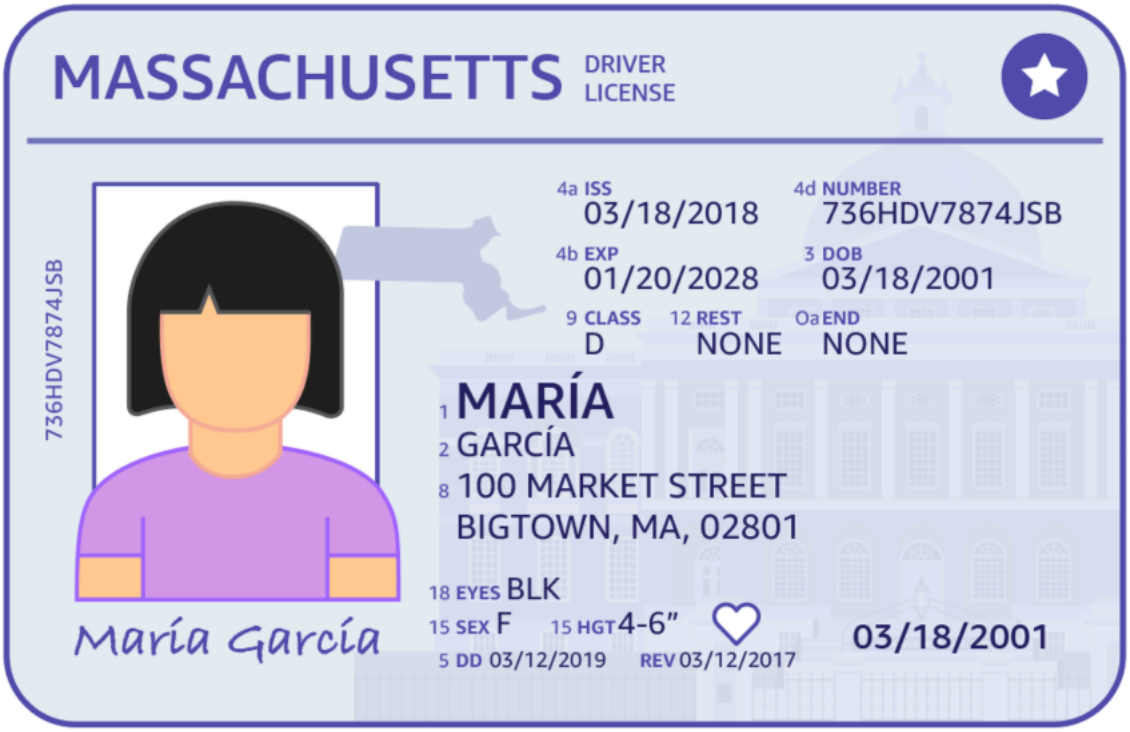
若要異步分析發票和收據，請使用StartExpenseAnalysis開始處理輸入文檔文件。若要獲取結果，請調用GetExpenseAnalysis。給定調用的結果StartExpenseAnalysis返回的GetExpenseAnalysis。如需詳細資訊和範例，請參閱 使用異步操作處理文檔。

分析身分文件

Amazon Textract 可以從護照、駕駛執照和美國政府使用 AnalyzeID API 頒發的其他身份文件中提取相關信息。藉助 Analysis ID，企業可以快速準確地從身份證中提取信息，如美國駕駛執照、州 ID 和具有不同模板或格式的護照。分析 ID API 返回兩類數據類型：

- ID 上提供的鍵值對，例如出生日期、簽發日期、ID 號、班級和限制。
- 文檔中可能沒有與其關聯的顯式密鑰的隱含字段，例如姓名、地址和頒發者。

響應中的關鍵名稱是標準化的。例如，如果您的駕駛執照顯示 LIC#（執照號碼），護照顯示「護照號碼」，則分析身份證響應將標準化密鑰與原始密鑰一起返回為「文件 ID」（例如 LIC#）。通過這種標準化，客戶可以輕鬆地將多個 ID 中的信息組合在一起，這些 ID 使用不同術語來實現



分析 ID 返回名為IdentityDocumentFields。這些都是JSON結構，其中包含兩段信息：規範化的類型和與類型關聯的值。這兩者都有一個信心得分。如需詳細資訊，請參閱 標識文件回應物件。

您可以使用同步操作來分析駕駛執照或護照。要分析這些文檔，請使用 AnalyzeID 操作並將身份文檔傳遞給它。AnalyzeID返回整組結果。如需詳細資訊，請參閱 使用 Amazon Textract 分析身份文檔。

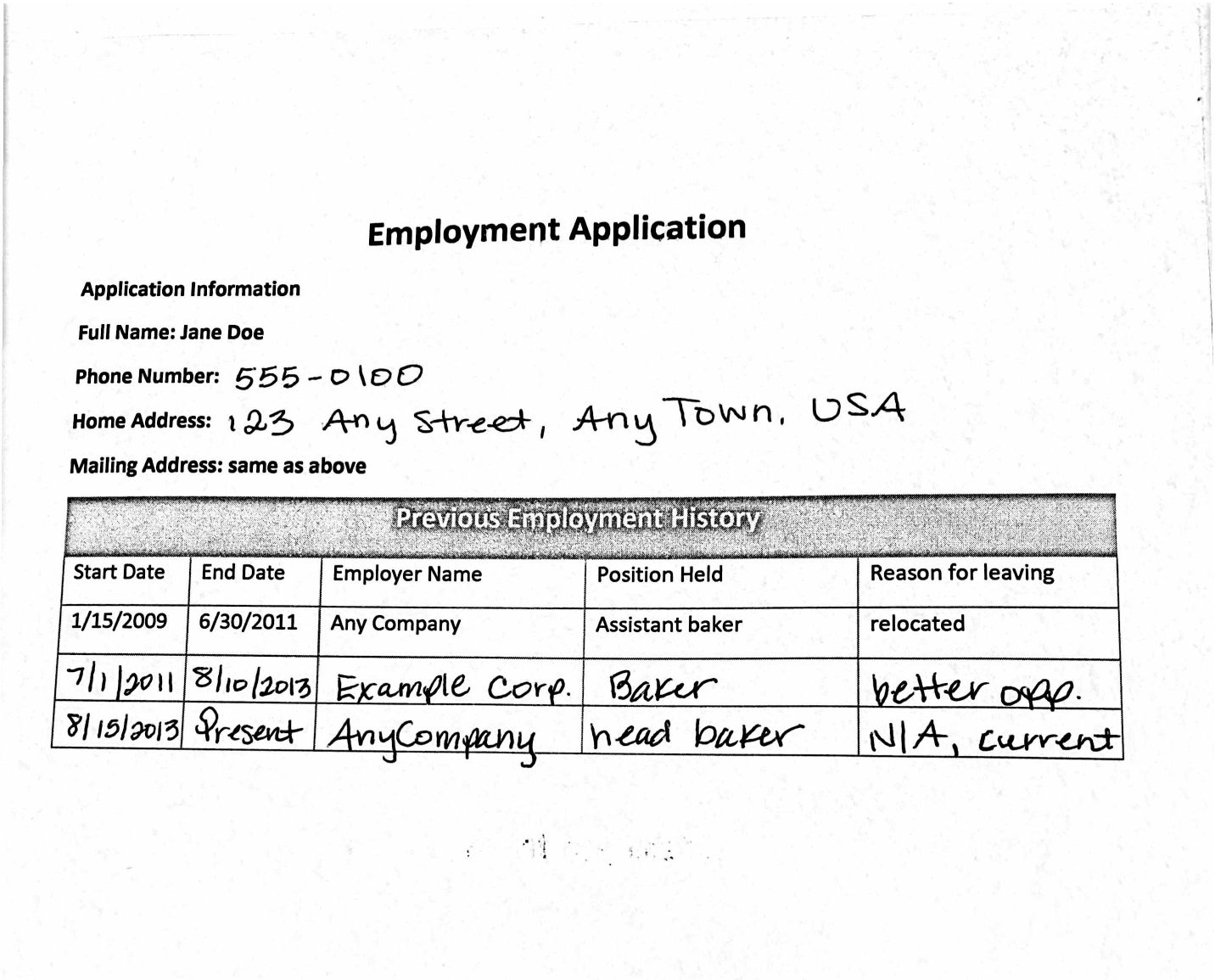
> **Note**
>
> 有些身份證件，例如駕駛執照，有兩面。您可以在同一個分析 ID API 請求中作為單獨的映像傳遞駕駛執照的正面和背面映像。

輸入文檔

Amazon Textract 操作的合適輸入是單頁或多頁文檔。一些示例包括法律文件、表單、ID 或信件。表單是一個帶有問題或提示用户提供答案的文檔。一些示例包括患者登記表、税務表或保險索賠表。

文檔可以採用 JPEG、PNG、PDF 格式或 TIFF 格式。使用 PDF 和 TIFF 格式文件，您可以處理多頁文檔。有關 Amazon Textract 如何將文檔表示為Block對象，請參閱文本檢測和文檔分析響應對象。

以下為可接受的輸入文件範例。



Employment Application

Application Information

Full Name: Jane Doe

Phone Number: 555-0100

Home Address: 123 Any Street, Any Town, USA

Mailing Address: same as above

Previous Employment History

Start Date	End Date	Employer Name	Position Held	Reason for leaving
1/15/2009	6/30/2011	Any Company	Assistant baker	relocated
7/1/2011	8/10/2013	Example Corp.	Baker	better opp.
8/15/2013	Present	AnyCompany	head baker	N/A, current

如需文件限制的資訊，請參亞馬遜文字中的硬性限制。

對於 Amazon Textract 同步操作，您可以使用存儲在 Amazon S3 存儲桶中的輸入文檔，也可以傳遞 base64 編碼的圖像字節。如需詳細資訊，請參閱 調用 Amazon Textract 同步操作。對於異步操作，您需要在 Amazon S3 存儲桶中提供輸入文檔。如需詳細資訊，請參閱 調用 Amazon Textract 異步操作。

Amazon Textract 應物件

Amazon Textract 操作根據運行的操作返回不同類型的對象。為了檢測文本和分析通用文檔，操作會返回一個 Block 對象。為了分析發票或收據，該操作將返回一個費用文檔對象。為了分析身份文檔，該操作將返回一個標識文檔字段對象。如需這些回應物件的詳細資訊，請參下列各節：

主題

文本檢測和文檔分析響應對象

當 Amazon Textract 處理文檔時，它會創建一個Block對象，用於檢測到或分析的文本。每個區塊都包含有關檢測到的商品的信息、其所在位置以及 Amazon Textract 對處理準確性的信心。

文檔由以下類型的Block物件。

- 頁面
- 文本的行和單詞
- 表單數據（鍵值對）
- 表格和單元格
- 選擇元素

塊的內容取決於您調用的操作。如果調用其中一個文本檢測操作，則返回檢測到的文本的頁面、行和單詞。如需詳細資訊，請參閱 偵測文字。如果調用其中一個文檔分析操作，則返回有關檢測到的頁面、鍵值對、表格、選擇元素和文本的信息。如需詳細資訊，請參閱 分析文檔。

一些Block物件欄位對於這兩類處理方式是相同的。例如，每個塊都有一個唯一的標識符。

如需的範例，示範如何使用Block對象，請參閱教學課程。

文檔佈局

Amazon Textract 返回一個文檔的表示形式，其中包含不同類型的Block在父子關係或鍵值對中鏈接的對象。還會返回提供文檔中頁數的元數據。以下為典型的Block類型 的物件PAGE。

```
{
    "Blocks": [
        {
            "Geometry": {
                "BoundingBox": {
                    "Width": 1.0,
                    "Top": 0.0,
```

```
                    "Left": 0.0,
                    "Height": 1.0
                },
                "Polygon": [
                    {
                        "Y": 0.0,
                        "X": 0.0
                    },
                    {
                        "Y": 0.0,
                        "X": 1.0
                    },
                    {
                        "Y": 1.0,
                        "X": 1.0
                    },
                    {
                        "Y": 1.0,
                        "X": 0.0
                    }
                ]
            },
            "Relationships": [
                {
                    "Type": "CHILD",
                    "Ids": [
                        "2602b0a6-20e3-4e6e-9e46-3be57fd0844b",
                        "82aedd57-187f-43dd-9eb1-4f312ca30042",
                        "52be1777-53f7-42f6-a7cf-6d09bdc15a30",
                        "7ca7caa6-00ef-4cda-b1aa-5571dfed1a7c"
                    ]
                }
            ],
            "BlockType": "PAGE",
            "Id": "8136b2dc-37c1-4300-a9da-6ed8b276ea97"
        }.....

    ],
    "DocumentMetadata": {
        "Pages": 1
    }
}
```

文件由一或多個PAGE區塊。每個頁面都包含頁面上檢測到的主要項目的子塊列表，例如文本行和表格。如需詳細資訊，請參閱 頁面。

您可以判斷Block對象，方法是檢查BlockType欄位。

一個Block物件包含相關Block中的物件Relationships字段，它是一個Relationship物件。一個Relationships數組的類型為「子」或「值」類型。CIDER 類型的數組用於列出作為當前塊子項的項目。例如，如果目前塊類型為 LINE，Relationships包含組成文本行的 WORD 塊的 ID 列表。VALUE 類型的陣列用於包含鍵/值對。您可以通過檢查Type欄位Relationship物件。

子塊沒有關於其父塊對象的信息。

對於顯示Block信息，請參閱使用同步操作處理文檔。

信賴度

Amazon Textract 操作會傳回 Amazon Textract 對於商品精確度的可信度百分比。要獲得信心，請使用Confidence欄位Block物件。值越大代表置信度越大。根據不同的場景，可信度低的檢測可能需要人類的視覺確認。

Geometry

Amazon Textract 操作（身份分析除外）會返回有關文檔頁面上檢測到商品位置的位置信息。若要獲取位置，請使用Geometry欄位Block物件。如需詳細資訊，請參閱「」文檔頁面上的項目位置

頁面

文件包含一或多個頁面。一個the section called “Block”類型 的物件PAGE存在於文檔的每個頁面。一個PAGE塊對象包含文檔頁面上檢測到的文本行、鍵值對和表的子 ID 的列表。

JSONPAGE塊結果類似如下。

```
{

    "Geometry": ....
    "Relationships": [
        {
            "Type": "CHILD",
            "Ids": [
```

```
                "2602b0a6-20e3-4e6e-9e46-3be57fd0844b", // Line - Hello, world.
                "82aedd57-187f-43dd-9eb1-4f312ca30042", // Line - How are you?
                "52be1777-53f7-42f6-a7cf-6d09bdc15a30",
                "7ca7caa6-00ef-4cda-b1aa-5571dfed1a7c"
            ]
        }
    ],
    "BlockType": "PAGE",
    "Id": "8136b2dc-37c1-4300-a9da-6ed8b276ea97"  // Page identifier
},
```

如果要對 PDF 格式的多頁文檔使用異步操作，則可以通過檢查Page欄位Block物件。掃描的圖像（JPEG、PNG、PDF 或 TIFF 格式的圖像）被視為單頁文檔，即使圖像上有多個文檔頁面。異步操作總是返回Page值為 1，用於掃描的圖像。

頁面總數會返回到Pages欄位DocumentMetadata。DocumentMetadata返回的每個列表Block由 Amazon Textract 操作返回的對象。

文本的行和單詞

由 Amazon Textract 操作返回的檢測到的文本將返回到the section called “Block”物件。這些對象表示在文檔頁面上檢測到的文本行或文本單詞。以下文本顯示了由多個單詞構成的兩行文本。

這是文字。

在兩個單獨的行。

檢測到的文本將在Text欄位Block物件。所以此BlockType字段確定文本是一行文本 (LINE) 還是單詞 (WORD)。一個字是一或多個 ISO 基本拉丁腳本字符，不以空格分隔。一個線是製表符分隔和連續單詞的字符串。

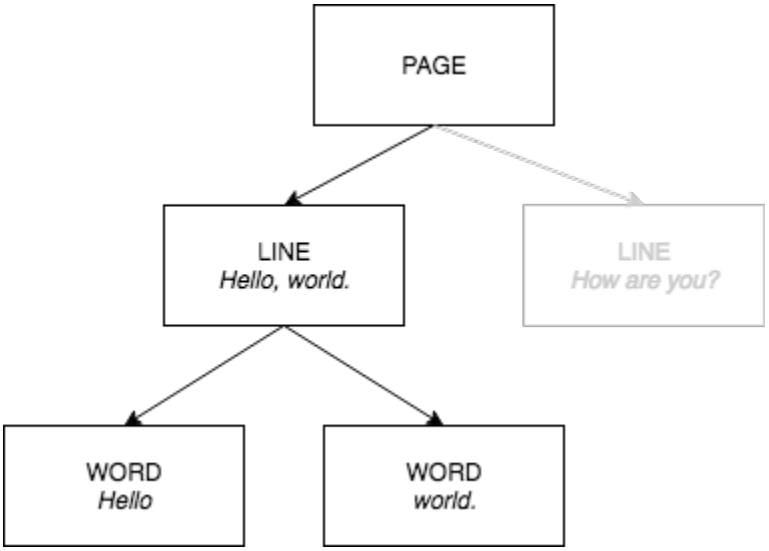
此外，Amazon Textract 將確定一段文本是手寫還是使用TextTypes欄位。這些分別作為手寫和印刷返回。

其他Block屬性對於所有塊類型（例如 ID、置信度和幾何信息）都是通用的。如需詳細資訊，請參閱 the section called “文本檢測和文檔分析響應對象”。

要僅檢測行和單詞，您可以使用DetectDocumentText或者StartDocumentTextDetection。如需詳細資訊，請參閱 偵測文字。要獲取檢測到的文本（行和單詞）以及有關它與文檔其他部分（如表格）相關的信息，您可以使用AnalyzeDocument或者StartDocumentAnalysis。如需詳細資訊，請參閱 分析文檔。

PAGE、LINE，以及WORD塊在父-子關係中彼此相關。一個PAGE塊是所有LINE塊文檔頁面上的對象。因為 LINE 可以有一個或多個單詞，Relationships數組存儲組成文本行的子 WORD 塊的 ID。

下圖顯示Hello world.在文字Hello world. 你怎麼樣表示為Block物件。



以下為來自DetectDocumentText當句子Hello world. 你怎麼樣被檢測到。第一個示例是文檔頁面的 JSON。請注意孩子 ID 如何使您能夠在文檔中導航。

```
{
    "Geometry": {...},
    "Relationships": [
        {
            "Type": "CHILD",
            "Ids": [
                "d7fbd604-d609-4d69-857d-247a3f591238", // Line - Hello, world.
                "b6c19a93-6493-4d8e-958f-853c8f7ca055" //  Line - How are you?
            ]
        }
    ],
    "BlockType": "PAGE",
    "Id": "56ec1d77-171f-4881-9852-2b5b7e761608"
},
```

以下是組成行「你好，世界」的 LINE 塊的 JSON：

```
{
    "Relationships": [
        {
```

```
            "Type": "CHILD",
            "Ids": [
                "7f97e2ca-063e-47a8-981c-8beee31afc01", // Word - Hello,
                "4b990aa0-af96-4369-b90f-dbe02538ed21"  // Word - world.
            ]
        }
    ],
    "Confidence": 99.63229370117188,
    "Geometry": {...},
    "Text": "Hello, world.",
    "BlockType": "LINE",
    "Id": "d7fbd604-d609-4d69-857d-247a3f591238"
},
```

以下是 WORD 塊的 JSONHello last：

```
{
    "Geometry": {...},
    "Text": "Hello,",
    "TextType": "PRINTED",
    "BlockType": "WORD",
    "Confidence": 99.74746704101562,
    "Id": "7f97e2ca-063e-47a8-981c-8beee31afc01"
},
```

最後的 JSON 是單詞的 WORD 塊世界。：

```
{
    "Geometry": {...},
    "Text": "world.",
    "TextType": "PRINTED",
    "BlockType": "WORD",
    "Confidence": 99.5171127319336,
    "Id": "4b990aa0-af96-4369-b90f-dbe02538ed21"
},
```

表單數據（鍵值對）

Amazon Textract 可以從文件中提取表單數據作為鍵/值對。例如，在以下的文字中，Amazon Textract 可以識別金鑰 (名稱:) 和一個值（安娜·卡羅萊納州。

名稱: 安娜·卡羅萊納州

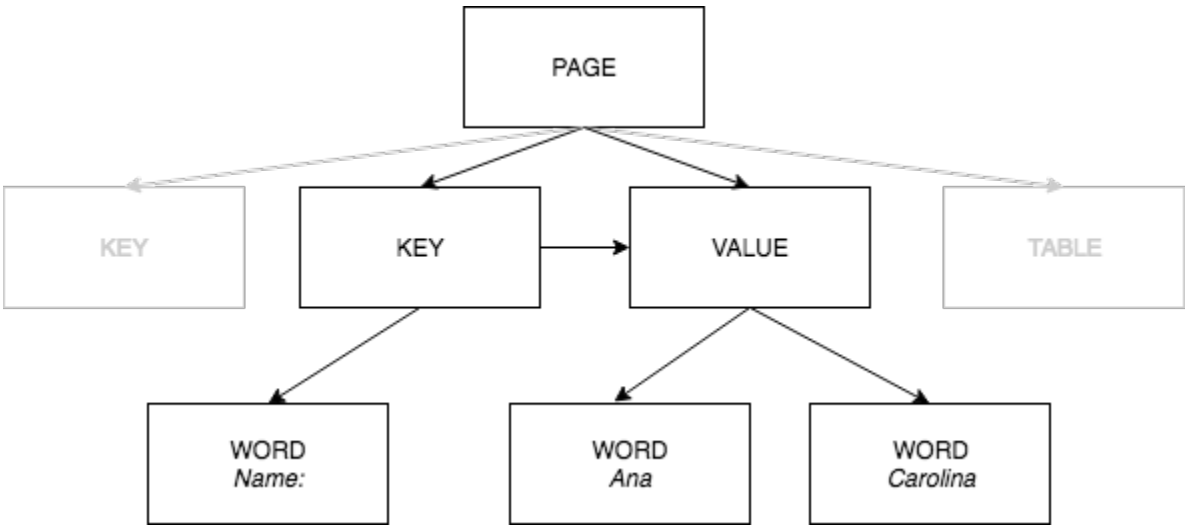
檢測到的金鑰值組返回為Block響應中的對象AnalyzeDocument和GetDocumentAnalysis。您可以使用FeatureTypes輸入參數來檢索關於鍵值對、表或兩者的信息。僅對於鍵值對，請使用值FORMS。如需範例，請參閱 從表單文檔中提取鍵值對。如需文件如何由Block對象，請參閱文本檢測和文檔分析響應對象。

類型為 KEY_VALUE_SET 的塊對象是用於存儲文檔中檢測到的鏈接文本項的信息的鍵或值塊對象的容器。您可以使用EntityType屬性來確定塊是 KEY 還是值。

- 一個鍵物件包含鏈接文字鍵的資訊。例如：名稱:。KEY 塊有兩個關係列表。VALUE 類型的關係是一個列表，其中包含與該鍵相關聯的 VALUE 塊的 ID。類型的關係是組成鍵文本的 WORD 塊的 ID 列表。
- 一個值物件包含與鍵相關的文字的資訊。在上述範例中，安娜·卡羅萊納州是鍵的值。名稱:。VALUE 塊與標識 WORD 塊的子塊列表存在關係。每個 WORD 塊包含組成值文本的單詞之一。一個VALUE物件也可以包含選取元素的資訊。如需詳細資訊，請參閱 選擇元素。

鍵值集的每個實例Block物件是 PAGE 的子項Block對象，該對應於當前頁面。

下圖顯示鍵/值對名稱: 安娜·卡羅萊納州表示為Block物件。



以下範例會示範鍵/值對名稱: 安娜·卡羅萊納州由 JSON 表示。

頁面塊具有類型為KEY_VALUE_SET對於文檔中檢測到的每個鍵和值塊。

```
{
    "Geometry": ....
    "Relationships": [
        {
            "Type": "CHILD",
```

```
            "Ids": [
                "2602b0a6-20e3-4e6e-9e46-3be57fd0844b",
                "82aedd57-187f-43dd-9eb1-4f312ca30042",
                "52be1777-53f7-42f6-a7cf-6d09bdc15a30", // Key - Name:
                "7ca7caa6-00ef-4cda-b1aa-5571dfed1a7c"  // Value - Ana Caroline
            ]
        }
    ],
    "BlockType": "PAGE",
    "Id": "8136b2dc-37c1-4300-a9da-6ed8b276ea97"  // Page identifier
},
```

下面的 JSON 顯示了密鑰塊（52be1777-53f7-4f6-4f6-a7c）與值塊有關係（7ca7Ca6-00EF -41a-51a-5571a7c）。它還有一個用於單詞塊的子塊（c734fca6-c4c4-415-b6c1-30f7510b72ee），其中包含密鑰的文本（名稱:。

```
{
    "Relationships": [
        {
            "Type": "VALUE",
            "Ids": [
                "7ca7caa6-00ef-4cda-b1aa-5571dfed1a7c"  // Value identifier
            ]
        },
        {
            "Type": "CHILD",
            "Ids": [
                "c734fca6-c4c4-415c-b6c1-30f7510b72ee"  // Name:
            ]
        }
    ],
    "Confidence": 51.55965805053711,
    "Geometry": ....,
    "BlockType": "KEY_VALUE_SET",
    "EntityTypes": [
        "KEY"
    ],
    "Id": "52be1777-53f7-42f6-a7cf-6d09bdc15a30"  //Key identifier
},
```

下面的 JSON 顯示值塊 7ca7caa6-00ef-4cda-b1aa-5571dfed1a7c 具有組成值文本的字塊的子 ID 列表 (安娜和卡羅萊納州。

```
{
    "Relationships": [
        {
            "Type": "CHILD",
            "Ids": [
                "db553509-64ef-4ecf-ad3c-bea62cc1cd8a", // Ana
                "e5d7646c-eaa2-413a-95ad-f4ae19f53ef3"  // Carolina
            ]
        }
    ],
    "Confidence": 51.55965805053711,
    "Geometry": ....,
    "BlockType": "KEY_VALUE_SET",
    "EntityTypes": [
        "VALUE"
    ],
    "Id": "7ca7caa6-00ef-4cda-b1aa-5571dfed1a7c" // Value identifier
}
```

下面的 JSON 顯示了Block單詞的對象名稱:、安娜，以及卡羅萊納州。

```
{
    "Geometry": {...},
    "Text": "Name:",
    "TextType": "PRINTED".
    "BlockType": "WORD",
    "Confidence": 99.56285858154297,
    "Id": "c734fca6-c4c4-415c-b6c1-30f7510b72ee"
},
 {
    "Geometry": {...},
    "Text": "Ana",
    "TextType": "PRINTED",
    "BlockType": "WORD",
    "Confidence": 99.52057647705078,
    "Id": "db553509-64ef-4ecf-ad3c-bea62cc1cd8a"
},
{
    "Geometry": {...},
    "Text": "Carolina",
    "TextType": "PRINTED",
    "BlockType": "WORD",
    "Confidence": 99.84207916259766,
```

```
    "Id": "e5d7646c-eaa2-413a-95ad-f4ae19f53ef3"
},
```

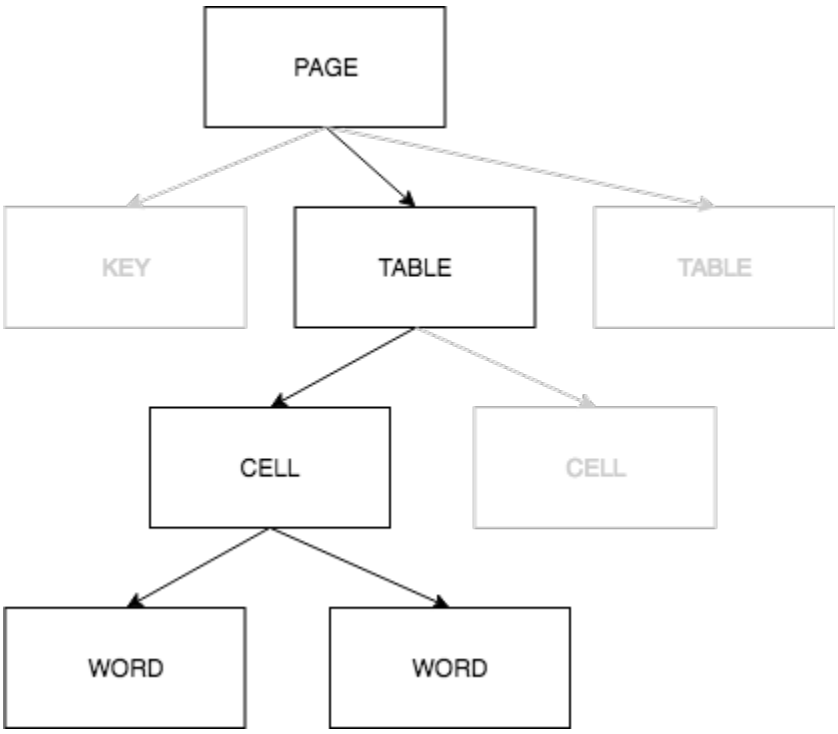
資料表

Amazon Textract 可以提取表格中的表格和單元格。例如，當在表單上檢測到下表時，Amazon Textract 會檢測到包含四個單元格的表格。

名稱	Address
安娜·卡羅萊納州	123 任何城市

檢測到的表返回為Block響應中的對象AnalyzeDocument和GetDocumentAnalysis。您可以使用FeatureTypes輸入參數來檢索關於鍵值對、表或兩者的信息。僅對於表格，請使用值TABLES。如需範例，請參閱 將資料表導出為 CSV 檔。如需文件如何由Block對象，請參閱文本檢測和文檔分析響應對象。

下圖顯示表中的單個單元格如何由Block物件。



單元格包含WORD用於檢測到的單詞的塊，以及SELECTION_ELEMENT塊的選擇元素（如複選框）。

以下是前面的表（包含四個單元格）的部分 JSON。

頁面塊對象具有 TABLE 塊和檢測到的每行文本的子塊 ID 的列表。

```
{
    "Geometry": {...},
    "Relationships": [
        {
            "Type": "CHILD",
            "Ids": [
                "f2a4ad7b-f21d-4966-b548-c859b84f66a4",   // Line - Name
                "4dce3516-ffeb-45e0-92a2-60770e9cb744",   // Line  - Address
                "ee506578-768f-4696-8f4b-e4917e429f50",   // Line - Ana Carolina
                "33fc7223-411b-4399-8a90-ccd3c5a2c196",   // Line  - 123 Any Town
                "3f9665be-379d-4ae7-be44-d02f32b049c2"    // Table
            ]
        }
    ],
    "BlockType": "PAGE",
    "Id": "78c3ce84-ae70-418e-add7-27058418adf6"
},
```

TABLE 塊包含表中單元格的子 ID 列表。TABLE 塊還包括文檔中表位置的幾何信息。下面的 JSON 顯示該表包含四個單元格，這些單元格在Ids陣列。

```
{
    "Geometry": {...},
    "Relationships": [
        {
            "Type": "CHILD",
            "Ids": [
                "505e9581-0d1c-42fb-a214-6ff736822e8c",
                "6fca44d4-d3d3-46ab-b22f-7fca1fbaaf02",
                "9778bd78-f3fe-4ae1-9b78-e6d29b89e5e9",
                "55404b05-ae12-4159-9003-92b7c129532e"
            ]
        }
    ],
    "BlockType": "TABLE",
    "Confidence": 92.5705337524414,
    "Id": "3f9665be-379d-4ae7-be44-d02f32b049c2"
},
```

表單元格的塊類型為 CELL。所以此Block對象包含與表中其他單元格相比的有關單元格位置的信息。它還包括單元格在文檔上位置的幾何信息。在上述範例中，505e9581-0d1c-42fb-a214-6ff736822e8c是包含單詞的單元格的子 ID名稱。以下示例是單元格的信息。

```
{
    "Geometry": {...},
    "Relationships": [
        {
            "Type": "CHILD",
            "Ids": [
                "e9108c8e-0167-4482-989e-8b6cd3c3653e"
            ]
        }
    ],
    "Confidence": 100.0,
    "RowSpan": 1,
    "RowIndex": 1,
    "ColumnIndex": 1,
    "ColumnSpan": 1,
    "BlockType": "CELL",
    "Id": "505e9581-0d1c-42fb-a214-6ff736822e8c"
},
```

每個單元格在表中都有一個位置，第一個單元格是 1,1。在上述範例中，值為名稱位於第 1 行，第 1 列。具有值的單元格123 任何城市位於第 2 行，第 2 列。單元格塊對象包含此信息在RowIndex和ColumnIndex和 欄位之間沒有任何差異。子列表包含包含單元格中的文本的 WORD 塊對象的 ID。列表中的單詞按照檢測到的順序，從單元格的左上角到單元格右下角。在上面的示例中，單元格具有一個子 ID，其值為 e9108c8e-0167-4482-989e-8c3e。下面的輸出是用於 ID 值為 E9108c8e-0167-4482-989-89-8cd3e 的字詞塊：

```
"Geometry": {...},
"Text": "Name",
"TextType": "Printed",
"BlockType": "WORD",
"Confidence": 99.81139373779297,
"Id": "e9108c8e-0167-4482-989e-8b6cd3c3653e"
},
```

選擇元素

Amazon Textract 可以檢測選擇元素，如選項按鈕（單選按鈕）和文檔頁面上的複選框。選擇元素可以在表單數據和表。例如，當在表單上檢測到下表時，Amazon Textract 會檢測到表格單元格中的複選框。

	同意	Neutral	不同意
良好的服務	☑	☐	☐
易於使用	☐	☑	☐
合理價格	☑	☐	☐

檢測到的選擇元素返回為Block響應中的對象AnalyzeDocument和GetDocumentAnalysis。

> **Note**
>
> 您可以使用FeatureTypes輸入參數來檢索關於鍵值對、表或兩者的信息。例如，如果對錶進行篩選，則響應包括在表中檢測到的選擇元素。在鍵值對中檢測到的選擇元素不包括在響應中。

有關選擇元素的信息包含在Block類型 的物件SELECTION_ELEMENT。若要判定可選元素的狀態，請使用SelectionStatus欄位SELECTION_ELEMENT區塊。狀態可能是選或者未選定。例如，值為SelectionStatus對於上一個圖像是選。

一個SELECTION_ELEMENT Block對象與鍵值對或表格單元格相關聯。一個SELECTION_ELEMENT Block對象包含選擇元素的邊界框信息Geometry欄位。一個SELECTION_ELEMENT Block對象不是PAGE Block物件。

表單數據（鍵值對）

鍵值對用於表示在表單上檢測到的選擇元素。所以此KEY塊包含選擇元素的文本。所以此VALUE塊包含選擇元素塊。下圖顯示選擇元素如何由the section called “Block”物件。

如需金鑰/值對的詳細資訊，請參表單數據（鍵值對）。

下面的 JSON 代碼段顯示了包含選擇元素 (男性 ☑。子代碼 (編號為標識字符串) 是包含選擇元素的文本塊的 ID (男性。該值的編號 (編號為二十四分之一) 是VALUE塊，其中包含SELECTION_ELEMENT塊物件。

```
{
    "Relationships": [
        {
            "Type": "VALUE",
            "Ids": [
                "24aaac7f-fcce-49c7-a4f0-3688b05586d4"  // Value containing Selection
 Element
            ]
        },
        {
            "Type": "CHILD",
            "Ids": [
                "bd14cfd5-9005-498b-a7f3-45ceb171f0ff"  // WORD - male
            ]
        }
    ],
    "Confidence": 94.15619659423828,
    "Geometry": {
        "BoundingBox": {
            "Width": 0.022914813831448555,
            "Top": 0.08072036504745483,
            "Left": 0.18966935575008392,
            "Height": 0.014860388822853565
        },
        "Polygon": [
            {
                "Y": 0.08072036504745483,
                "X": 0.18966935575008392
            },
            {
                "Y": 0.08072036504745483,
                "X": 0.21258416771888733
            },
            {
                "Y": 0.09558075666427612,
                "X": 0.21258416771888733
            },
            {
                "Y": 0.09558075666427612,
```

```
                "X": 0.18966935575008392
            }
        ]
    },
    "BlockType": "KEY_VALUE_SET",
    "EntityTypes": [
        "KEY"
    ],
    "Id": "a118dc43-d5f7-49a2-a20a-5f876d9ffd79"
}
```

下面的 JSON 代碼片段是單詞男性。WORD 塊也有一個父線塊。

```
{
    "Geometry": {
        "BoundingBox": {
            "Width": 0.022464623674750328,
            "Top": 0.07842985540628433,
            "Left": 0.18863198161125183,
            "Height": 0.01617223583161831
        },
        "Polygon": [
            {
                "Y": 0.07842985540628433,
                "X": 0.18863198161125183
            },
            {
                "Y": 0.07842985540628433,
                "X": 0.2110965996980667
            },
            {
                "Y": 0.09460209310054779,
                "X": 0.2110965996980667
            },
            {
                "Y": 0.09460209310054779,
                "X": 0.18863198161125183
            }
        ]
    },
    "Text": "Male",
    "BlockType": "WORD",
    "Confidence": 54.06439208984375,
```

```
    "Id": "bd14cfd5-9005-498b-a7f3-45ceb171f0ff"
},
```

值塊有一個子項（編號為 f2f5e8d-e73a-4e99-a095-053），它是選擇元素塊。

```
{
    "Relationships": [
        {
            "Type": "CHILD",
            "Ids": [
                "f2f5e8cd-e73a-4e99-a095-053acd3b6bfb"  // Selection element
            ]
        }
    ],
    "Confidence": 94.15619659423828,
    "Geometry": {
        "BoundingBox": {
            "Width": 0.017281491309404373,
            "Top": 0.07643391191959381,
            "Left": 0.2271782010793686,
            "Height": 0.026274094358086586
        },
        "Polygon": [
            {
                "Y": 0.07643391191959381,
                "X": 0.2271782010793686
            },
            {
                "Y": 0.07643391191959381,
                "X": 0.24445968866348267
            },
            {
                "Y": 0.10270800441503525,
                "X": 0.24445968866348267
            },
            {
                "Y": 0.10270800441503525,
                "X": 0.2271782010793686
            }
        ]
    },
    "BlockType": "KEY_VALUE_SET",
    "EntityTypes": [
```

```
        "VALUE"
    ],
    "Id": "24aaac7f-fcce-49c7-a4f0-3688b05586d4"
},
}
```

以下 JSON 是選擇元素塊。的值SelectionStatus指示已選取核取方塊。

```
{
    "Geometry": {
        "BoundingBox": {
            "Width": 0.020316146314144135,
            "Top": 0.07575977593660355,
            "Left": 0.22590067982673645,
            "Height": 0.027631107717752457
        },
        "Polygon": [
            {
                "Y": 0.07575977593660355,
                "X": 0.22590067982673645
            },
            {
                "Y": 0.07575977593660355,
                "X": 0.2462168186903
            },
            {
                "Y": 0.1033908873796463,
                "X": 0.2462168186903
            },
            {
                "Y": 0.1033908873796463,
                "X": 0.22590067982673645
            }
        ]
    },
    "BlockType": "SELECTION_ELEMENT",
    "SelectionStatus": "SELECTED",
    "Confidence": 74.14942932128906,
    "Id": "f2f5e8cd-e73a-4e99-a095-053acd3b6bfb"
}
```

表格

Amazon Textract 可以檢測表格單元格中的選擇元素。例如，下表中的單元格具有複選框。

	同意	Neutral	不同意
良好的服務	☑	☐	☐
易於使用	☐	☑	☐
合理價格	☑	☐	☐

一個CELL塊可以包含子項SELECTION_ELEMENT選擇元素的對象，以及子WORD塊，用於檢測到的文本。

如需表的詳細資訊，請參資料表。

該表Block對象看起來類似於此。

```
{
    "Geometry": {.....},
    "Relationships": [
        {
            "Type": "CHILD",
            "Ids": [
                "652c09eb-8945-473d-b1be-fa03ac055928",
                "37efc5cc-946d-42cd-aa04-e68e5ed4741d",
                "4a44940a-435a-4c5c-8a6a-7fea341fa295",
                "2de20014-9a3b-4e26-b453-0de755144b1a",
                "8ed78aeb-5c9a-4980-b669-9e08b28671d2",
                "1f8e1c68-2c97-47b2-847c-a19619c02ca9",
                "9927e1d1-6018-4960-ac17-aadb0a94f4d9",
                "68f0ed8b-a887-42a5-b618-f68b494a6034",
                "fcba16e0-6bd7-4ea5-b86e-36e8330b68ea",
                "2250357c-ae34-4ed9-86da-45dac5a5e903",
                "c63ad40d-5a14-4646-a8df-2d4304213dbc",   // Cell
                "2b8417dc-e65f-4fcd-aa0f-61a23f1e8cb0",
                "26c62932-72f0-4dc2-9893-1ae27829c060",
                "27f291cc-abf4-4c23-aa24-676abe99cb1e",
                "7e5ce028-1bcd-4d9f-ad42-15ac181c5b47",
```

```
                "bf32e3d2-efa2-4fc1-b09b-ab9cc52ff734"
            ]
        }
    ],
    "BlockType": "TABLE",
    "Confidence": 99.99993896484375,
    "Id": "f66eac36-2e74-406e-8032-14d1c14e0b86"
}
```

儲存格BLOCK對象（編號為複選框的單元格）良好的服務看起來如下。它包括一個孩子Block（編號 = 二十二分之二）SELECTION_ELEMENT Block物件。

```
{
    "Geometry": {.....},
    "Relationships": [
        {
            "Type": "CHILD",
            "Ids": [
                "26d122fd-c5f4-4b53-92c4-0ae92730ee1e"  // Selection Element
            ]
        }
    ],
    "Confidence": 79.741689682006836,
    "RowSpan": 1,
    "RowIndex": 3,
    "ColumnIndex": 3,
    "ColumnSpan": 1,
    "BlockType": "CELL",
    "Id": "c63ad40d-5a14-4646-a8df-2d4304213dbc"
}
```

選擇元素Block對象如下所示。的值SelectionStatus指示已選取核取方塊。

```
{
    "Geometry": {.......},
    "BlockType": "SELECTION_ELEMENT",
    "SelectionStatus": "SELECTED",
    "Confidence": 88.79517364501953,
    "Id": "26d122fd-c5f4-4b53-92c4-0ae92730ee1e"
}
```

發票和收據響應對象

當您向分析費用 API 提交發票或收據時，它會返回一系列費用文檔對象。每個費用文檔進一步分隔為LineItemGroups和SummaryFields。大多數發票和收據包含諸如供應商名稱、接收編號、接收日期或總金額等信息。分析費用將此信息返回SummaryFields。收據和發票還包含有關購買物料的詳細信息。分析費用 API 將此信息返回LineItemGroups。所以此ExpenseIndex字段唯一標識支出，並將相應的SummaryFields和LineItemGroups檢測到該費用。

分析支出響應中最精細的數據級別包括Type、ValueDetection，以及LabelDetection(選用)。各個實體是：

- 類型：指在高級別上檢測到的信息類型。
- 標籤檢測：引用文檔文本中關聯值的標籤。LabelDetection是可選的，並且僅在寫入標籤時返回。
- 值檢測：指返回的標籤或類型的值。

分析費用 API 還會檢測到ITEM、QUANTITY，以及PRICE作為規範化字段。如果收據圖片上的單項商品中還有其他文本，例如 SKU 或詳細描述，則該文本將包含在 JSON 中，EXPENSE_ROW，如下例所示：

```
        {
                            "Type": {
                                "Text": "EXPENSE_ROW",
                                "Confidence": 99.95216369628906
                            },
                            "ValueDetection": {
                                "Text": "Banana 5 $2.5",
                                "Geometry": {
                                  …
                                },
                                "Confidence": 98.11214447021484
                            }
```

上面的示例顯示了 AnalyzeFesort API 如何返回包含有關以 2.5 美元售出的 5 香蕉的行項目信息的收據上的整個行。

類型

以下為鍵/值對的標準或規範化類型的範例：

```
            {
                 "PageNumber": 1,
                 "Type": {
                     "Text": "VENDOR_NAME",
                     "Confidence": 70.0
                 },
                 "ValueDetection": {
                     "Geometry": { ... },
                     "Text": "AMAZON",
                     "Confidence": 87.89806365966797
                 }
             }
```

收據沒有明確列出「供應商名稱」。但是，分析費用 API 將文檔識別為收據，並將值「AMAZON」分類為類型VENDOR_NAME。

標籤檢測

以下是在客户文檔頁面上顯示的文本示例：

```
            {
                 "PageNumber": 1,
                 "Type": {
                     "Text": "OTHER",
                     "Confidence": 70.0
                 },
                 "LabelDetection": {
                     "Geometry": { ... },
                     "Text": "CASHIER",
                     "Confidence": 88.19171142578125
                 },
                 "ValueDetection": {
                     "Geometry": { ... },
                     "Text": "Mina",
                     "Confidence": 87.89806365966797
                 }
```

```
        }
```

示例文件包含「出納米娜」。分析費用 API 提取了原樣值並將其返回到LabelDetection。對於收據中沒有明確顯示「鑰匙」的「供應商名稱」等隱含值,LabelDetection將不會包含在分析費用元素中。在這種情況下，分析費用 API 不會返回LabelDetection。

值檢測

以下範例示範鍵/值對的「值」。

```
        {
             "PageNumber": 1,
             "Type": {
                 "Text": "OTHER",
                 "Confidence": 70.0
             },
             "LabelDetection": {
                 "Geometry": { ... },
                 "Text": "CASHIER",
                 "Confidence": 88.19171142578125
             },
             "ValueDetection": {
                 "Geometry": { ... },
                 "Text": "Mina",
                 "Confidence": 87.89806365966797
             }
        }
```

在這個例子中，文檔包含「收銀員米娜」。分析費用 API 檢測到出納值為 Mina，並在ValueDetection。

標識文件回應物件

當您向 AnalyzeID API 提交身份文檔時，它會返回一系列IdentityDocumentField物件。這些對象中的每個都包含Type，以及Value。Type記錄亞 Amazon Textract 檢測到的標準化字段，Value記錄與規範化字段關聯的文本。

以下是IdentityDocumentField, 為簡潔而縮短.

```
{
    "DocumentMetadata": {
        "Pages": 1
    },
    "IdentityDocumentFields": [
        {
            "Type": {
                "Text": "first name"
            },
            "ValueDetection": {
                "Text": "jennifer",
                "Confidence": 99.99908447265625
            }
        },
        {
            "Type": {
                "Text": "last name"
            },
            "ValueDetection": {
                "Text": "sample",
                "Confidence": 99.99758911132812
            }
        },
```


這些是從較長響應中剪切的標識文檔字段的兩個示例。檢測到的類型和該類型的值之間存在分離。這裏，它分別為名字和姓氏。此結構重複包含的所有信息。如果某個類型未被識別為標準化字段，它將被列為「其他」。

以下是駕駛執照標準化字段的列表：

- 名字
- 姓氏
- 中間名字
- 後綴
- 地址中的城市
- 地址中的郵政編碼
- 狀態在地址中

- 縣
- 文檔編號
- 過期日期
- 出生日期
- 狀態名稱
- 發行日期
- class
- 限制
- 背書
- ID 類型
- 退伍軍人
- address

以下是美國護照的標準化字段列表：

- 名字
- 姓氏
- 中間名字
- 文檔編號
- 過期日期
- 出生日期
- 出生地
- 發行日期
- ID 類型

文檔頁面上的項目位置

Amazon Textract 操作會返回文檔頁面上找到的商品的位置和幾何形狀。DetectDocumentText和GetDocumentTextDetection返回線和單詞的位置和幾何，而AnalyzeDocument和GetDocumentAnalysis返回鍵/值組、表格、單元格和選擇元素的位置和幾何。

若要確定項目在文件頁面上的位置，請使用週框 (Geometry ）由 Amazon Textract 操作返回的信息Block物件。所以此Geometry物件包含兩類檢測到的項目的位置和幾何資訊：

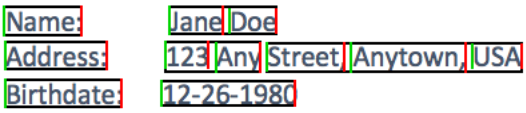
- 軸對齊BoundingBox物件，該物件包含左上方座標以及項目的寬度和高度。
- 描述項目輪廓的多邊形對象，指定為Point對象包含X(水平軸) 和Y（垂直軸）每個點的文檔頁面座標。

JSONBlock物件看起來類似以下的內容。請注意BoundingBox和Polygon和 欄位之間沒有任何差異。


```
{
    "Geometry": {
        "BoundingBox": {
            "Width": 0.053907789289951324,
            "Top": 0.08913730084896088,
            "Left": 0.11085548996925354,
            "Height": 0.013171200640499592
        },
        "Polygon": [
            {
                "Y": 0.08985357731580734,
                "X": 0.11085548996925354
            },
            {
                "Y": 0.08913730084896088,
                "X": 0.16447919607162476
            },
            {
                "Y": 0.10159222036600113,
                "X": 0.16476328670978546
            },
            {
                "Y": 0.10230850428342819,
                "X": 0.11113958805799484
            }
        ]
    },
    "Text": "Name:",
    "TextType": "PRINTED",
    "BlockType": "WORD",
    "Confidence": 99.56285858154297,
    "Id": "c734fca6-c4c4-415c-b6c1-30f7510b72ee"
```

```
},
```

可以使用幾何信息在檢測到的項目周圍繪製邊界框。對於使用BoundingBox和Polygon信息，以在每個單詞的開頭和結尾處繪製線和垂直線周圍的框，請參閱使用 Amazon Textract 檢測文檔文本。範例輸出類似如下。



Bounding Box (週框方塊)

邊框 (BoundingBox) 具有以下屬性：

- 高度 — 週框方塊的高度，以整體文檔頁面高度的比例表示。
- 左邊 — 左上方方塊的 X 座標，以整體文檔頁面寬度的比例表示。
- 上方 — 左上方方方塊的 Y 座標，以整體文檔頁面高度的比例表示。
- 寬度 — 週框方塊的寬度，以整體文件頁面寬度的比例表示。

每個 BoundingBox 屬性都有一個介於 0 和 1 的值。值為整體影像寬度的比例 (適用於Left和Width) 或高度 (適用於Height和Top。例如，如果輸入影像為 700 x 200 像素，而周框方塊的左上方座標為 (350,50) 像素，則 API 會傳回Left值為 0.5 (350/700) 和一個Top值為 0.25 (50/200).

下圖顯示每個 BoundingBox 屬性涵蓋的文檔頁面範圍。

若要顯示正確位置和大小的定框方塊，您必須將 BoundingBox 值乘以文檔頁面寬度或高度 (取決於您想要的值) 來取得像素值。您可以使用像素值來顯示週框方塊。一個示例是使用 608 像素寬 x 588 像素高度的文檔頁面，以及分析文本的以下邊界框值：

```
BoundingBox.Left: 0.3922065
BoundingBox.Top: 0.15567766
BoundingBox.Width: 0.284666
BoundingBox.Height: 0.2930403
```

文字週框方塊的位置 (以像素表示) 的計算方式如下：

```
Left coordinate = BoundingBox.Left (0.3922065) * document page width (608)
= 238
```

```
Top coordinate = BoundingBox.Top (0.15567766) * document page height (588)
= 91

Bounding box width = BoundingBox.Width (0.284666) * document page width
(608) = 173

Bounding box height = BoundingBox.Height (0.2930403) * document page height
(588) = 172
```

您可以使用這些值來顯示分析的文字周圍的週框方塊。以下 Java 和 Python 範例演示如何顯示週框方塊。

Java

```
   public void ShowBoundingBox(int imageHeight, int imageWidth, BoundingBox box,
Graphics2D g2d) {

        float left = imageWidth * box.getLeft();
        float top = imageHeight * box.getTop();

        // Display bounding box.
        g2d.setColor(new Color(0, 212, 0));
        g2d.drawRect(Math.round(left / scale), Math.round(top / scale),
                Math.round((imageWidth * box.getWidth()) / scale),
Math.round((imageHeight * box.getHeight())) / scale);

   }
```

Python

這個 Python 示例採用了`response`返回的DetectDocumentTextAPI 操作。

```
def process_text_detection(response):

    # Get the text blocks
    blocks = response['Blocks']
    width, height = image.size
    draw = ImageDraw.Draw(image)
    print('Detected Document Text')

    # Create image showing bounding box/polygon the detected lines/text
```

```
    for block in blocks:

        draw = ImageDraw.Draw(image)

        if block['BlockType'] == "LINE":
            box=block['Geometry']['BoundingBox']
            left = width * box['Left']
            top = height * box['Top']
            draw.rectangle([left,top, left + (width * box['Width']), top +(height *
box['Height'])],outline='black')

    # Display the image
    image.show()

    return len(blocks)
```

多邊形

返回的面AnalyzeDocument是Point物件。每個Point在文檔頁面上的特定位置具有 X 和 Y 座標。與「BoundingBox」座標一樣，面座標被歸一化為文檔寬度和高度，並且介於 0 和 1 之間。

您可以使用多邊形數組中的點在Block物件。計算文檔頁面上每個多邊形點的位置，方法是使用BoundingBoxes。將 X 座標乘以文檔頁面寬度，然後將 Y 座標乘以文檔頁面高度。

以下範例示範如何顯示多邊形的垂直線。

```
   public void ShowPolygonVerticals(int imageHeight, int imageWidth, List <Point>
points, Graphics2D g2d) {

       g2d.setColor(new Color(0, 212, 0));
       Object[] parry = points.toArray();
       g2d.setStroke(new BasicStroke(2));

       g2d.drawLine(Math.round(((Point) parry[0]).getX() * imageWidth),
               Math.round(((Point) parry[0]).getY() * imageHeight),
Math.round(((Point) parry[3]).getX() * imageWidth),
               Math.round(((Point) parry[3]).getY() * imageHeight));

       g2d.setColor(new Color(255, 0, 0));
       g2d.drawLine(Math.round(((Point) parry[1]).getX() * imageWidth),
               Math.round(((Point) parry[1]).getY() * imageHeight),
Math.round(((Point) parry[2]).getX() * imageWidth),
```

```
                Math.round(((Point) parry[2]).getY() * imageHeight));

    }
```

Amazon Textract 入門

本節主題可協助您開始使用 Amazon Textract。如果您是 Amazon Textract 新手，建議您先回顧Amazon Textract 工作方式。

您可以通過使用 Amazon Textract 控制台中的演示來嘗試 API。如需詳細資訊，請參閱「」https://console.aws.amazon.com/textract/。

主題

- 步驟 1：設定 AWS 帳户並建立 IAM 使用者
- 步驟 2：設定AWS CLI和AWS開發套件
- 步驟 3：使用入門AWS CLI和AWS軟件開發工具包 API

步驟 1：設定 AWS 帳户並建立 IAM 使用者

初次使用 Amazon Textract 之前，請先完成以下任務：

1. 註冊 AWS 帳號：.
2. 建立 IAM 使用者.

註冊 AWS 帳號：

註冊 Amazon Web Services (AWS) 時，您的 AWS 帳户會自動註冊 AWS 的所有已發佈服務。您只需針對所使用的服務付費。

使用 Amazon Textract 時，您僅需按使用的資源量付費。如需 Amazon Textract 使用費率的詳細資訊，請參Amazon Textract 定價。如果您是 AWS 新客户，可免費開始使用 Amazon Textract。如需詳細資訊，請參閱 AWS 免費用量方案。

若您已有 AWS 帳戶，請跳至下一個任務。如果您沒有 AWS 帳戶，請執行下列程序中的步驟建立一個帳戶。

建立 AWS 帳戶

1. 開啟 https://portal.aws.amazon.com/billing/signup。
2. 請遵循線上指示進行。

部分註冊程序需接收來電，並在電話鍵盤輸入驗證碼。

請記下您的 AWS 帳戶 ID，因為您下一個任務會需要此 ID。

建立 IAM 使用者

AWS 服務，例如 Amazon Textract，會在您進行存取時，要求您提供登入資料。這樣服務可以確定您是否有權存取該服務所擁有的資源。主控台需要您的密碼。您可以建立 AWS 帳戶的存取金鑰，用以存取 AWS CLI 或 API。不過，不建議您透過使用 AWS 帳戶的登入資料來存取 AWS。我們建議您改進行下列動作：

- 使用AWS Identity and Access Management(IAM) 來建立 IAM 使用者。
- 新增使用者到具備管理許可的 IAM 羣組.

您可以使用特殊 URL 與該 IAM 使用者的登入資料來存取 AWS。

如果您已註冊 AWS，但是尚未為自己建立 IAM 使用者，可以使用 IAM 主控台加以建立。請遵循程序在您的帳戶中建立 IAM 使用者。

建立 IAM 使用者並登入主控台

1. 在您的 AWS 帳戶中建立一個擁有管理員權限的 IAM 使用者。如需說明，請參閱「」建立您的第一個 IAM 使用者和管理員組中的IAM User Guide。
2. 做為 IAM 使用者，請使用特殊 URL 登入 AWS Management Console。如需詳細資訊，請參閱「」使用者如何登入您的帳戶中的IAM User Guide。

> **Note**
>
> 具有管理員許可的 IAM 使用者可以不受限制地存取AWS服務在您的帳戶中。
> 本指南中的代碼範例假設您有一個使用者`AmazonTextractFullAccess`許可。`AmazonS3ReadOnlyAccess`是存取存放在 Amazon S3 存儲體中的檔案的範例所必需的。根據您的安全要求，您可能希望使用限於這些許可的 IAM 羣組。如需詳細資訊，請參閱「」建立 IAM 羣組。

如需 IAM 的詳細資訊，請參閱下列各項：

- AWS Identity and Access Management (IAM)
- 入門
- IAM 使用者指南

後續步驟

步驟 2：設定AWS CLI和AWS開發套件

步驟 2：設定AWS CLI和AWS開發套件

以下步驟說明如何安裝本文件中使用的範例的 AWS Command Line Interface (AWS CLI) 和 AWS 開發套件。

有幾種不同的方式來驗證 AWS 開發套件呼叫。本指南中的範例假設您使用預設的登入資料設定檔來呼叫 AWS CLI 命令和 AWS 開發套件 API 操作。您的默認憑據將跨服務運行，因此，如果您已配置憑據，則無需再次配置此操作。但是，如果要為此服務創建另一組憑據，則可以創建名稱配置文件。如需建立檔的詳細資訊，請參命名設定檔。

如需可用的AWS區域，請參閱區域與端點中的Amazon Web Services 一般參考。

設定 AWS CLI 和 AWS 開發套件

1. 下載並安裝您要使用的 AWS CLI 和 AWS 開發套件。本指南提供AWS CLI、Java 和 Python。如需其他 AWS 開發套件的相關資訊，請參閱適用於 Amazon Web Services 的工具。

   - AWS CLI
   - AWS SDK for Java
   - AWS SDK for Python (Boto3)

2. 為您在建立 IAM 使用者。

   a. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
   b. 在導覽窗格中，選擇 Users (使用者)。
   c. 選擇您在建立 IAM 使用者。
   d. 選擇 Security credentials (安全憑證) 索引標籤。
   e. 選擇 Create access key (建立新的存取金鑰)。然後選擇 Download .csv file (下載 .csv 檔案)，將存取金鑰 ID 和私密存取金鑰儲存至電腦上的 CSV 檔案。將檔案存放在安全位置。在

關閉此對話方塊後，您將無法再次存取該私密存取金鑰。在您下載 CSV 檔後，選擇Close (關閉)。

3. 在您本機系統的 AWS 登入資料設定檔中設定登入資料，該設定檔位置於：

   - `~/.aws/credentials`在 Linux、macOS 或 Unix。
   - `C:\Users\USERNAME\.aws\credentials`Windows 上的。

   所以此`.aws`文件夾在您的 AWS 實例的第一次初始配置之前不存在。首次使用 CLI 配置憑據時，將創建此文件夾。如需 AWS 登入資料的詳細資訊，請參組態與登入資料檔案設定。

   此檔案應該包含下列格式的行：

   ```
   [default]
   aws_access_key_id = your_access_key_id
   aws_secret_access_key = your_secret_access_key
   ```

   替換您的存取金鑰 ID 和私密存取金鑰您的訪問密鑰 ID和您的祕密訪問密鑰。

4. 在 AWS 中設定預設 AWS 區域`config`檔案，該檔案位於：

   - `~/.aws/config`在 Linux、macOS 或 Unix。
   - `C:\Users\USERNAME\.aws\config`Windows 上的。

   所以此`.aws`文件夾在您的 AWS 實例的第一次初始配置之前不存在。首次使用 CLI 配置憑據時，將創建此文件夾。如需 AWS 登入資料的詳細資訊，請參組態與登入資料檔案設定。

   此檔案應該包含下列行：

   ```
   [default]
   region = your_aws_region
   ```

   替換您想要的 AWS 區域 (例如，「us-west-2")您的區域。

   > **Note**
   >
   > 如果您不選擇區域，則預設使用 us-east-1。

後續步驟

步驟 3：使用入門AWS CLI和AWS軟件開發工具包 API

步驟 3：使用入門AWS CLI和AWS軟件開發工具包 API

在您設置AWS CLI和AWS您可以建置使用 Amazon Textract 的應用程式。下列主題說明如何開始使用 Amazon Textract。

- 使用 Amazon Textract 分析文檔文本

將 AWS CLI 範例格式化

本指南中的 AWS CLI 範例已格式化為 Linux 作業系統。若要搭配 Microsoft Windows 使用範例，您需要變更 `--document` 參數的 JSON 格式，並將換行符號從反斜線 (\) 變更為插入號 (^)。如需 JSON 格式的詳細資訊，請參閱指定 AWS 命令列界面的參數值。

使用同步操作處理文檔

Amazon Textract 可以檢測和分析以 JPEG、PNG、PDF 和 TIFF 格式提供的單頁文檔中的文本。操作是同步的，會以接近即時的速度返回結果。如需有關文件的詳細資訊，請參閱 文本檢測和文檔分析響應對象。

本節介紹瞭如何使用 Amazon Textract 同步檢測和分析單頁文檔中的文本。要檢測和分析多頁文檔中的文本，或異步檢測 JPEG 和 PNG 文檔，請參閱使用異步操作處理文檔。

您可以將 Amazon Textract 同步操作用於以下目的：

- 文本檢測 — 您可以通過使用DetectDocumentTextoperation. 如需詳細資訊，請參閱 偵測文字。
- 文本分析 — 您可以通過使用AnalyzeDocumentoperation. 如需詳細資訊，請參閱 分析文檔。
- 發票和收據分析 — 您可以使用 AnalyzeFesent 操作確定單頁發票上檢測到的文本或收據之間的財務關係。如需詳細資訊，請參閱「」分析發票和收款
- 身份證件分析 — 您可以分析美國政府頒發的身份證件，並提取信息以及身份證件上常見類型的信息。如需更多詳細資訊，請參閱。分析身分文件。

主題

- 調用 Amazon Textract 同步操作
- 使用 Amazon Textract 檢測文檔文本
- 使用 Amazon Textract 分析文檔文本
- 使用 Amazon Textract 分析發票和收據
- 使用 Amazon Textract 分析身份文檔

調用 Amazon Textract 同步操作

Amazon Textract 操作處理存儲在本地文件系統中的文檔圖像，或存儲在 Amazon S3 存儲桶中的文檔圖像。指定輸入文檔所在的位置，方法是使用Document輸入參數。文檔圖像可以採用 PNG、JPEG、PDF 格式或 TIFF 格式。同步操作的結果會立即返回，不會存儲以供檢索。

如需完整範例，請參使用 Amazon Textract 檢測文檔文本。

要求

下面介紹了請求在 Amazon Textract 中的工作原理。

作為圖像字節傳遞的文檔

您可以通過將文檔圖像作為 base64 編碼字節數組傳遞給 Amazon Textract 操作。一個示例是從本地文件系統加載的文檔圖像。如果您使用的是AWS軟件開發工具包調用 Amazon Textract API 操作。

圖像字節在`Bytes`欄位`Document`輸入參數。以下示例顯示了 Amazon Textract 操作的輸入 JSON，該操作將圖像字節傳遞到`Bytes`輸入參數。

```
{
    "Document": {
        "Bytes": "/9j/4AAQSk....."
    }
}
```

> **Note**
>
> 如果您使用AWS CLI，您無法將圖像字節傳遞給 Amazon Textract 操作。相反，您必須參考存放在 Amazon S3 儲存貯體中的影像。

以下 Java 代碼演示如何從本機檔案系統載入影像，並呼叫 Amazon Textract 操作。

```
String document="input.png";

ByteBuffer imageBytes;
try (InputStream inputStream = new FileInputStream(new File(document))) {
    imageBytes = ByteBuffer.wrap(IOUtils.toByteArray(inputStream));
}
AmazonTextract client = AmazonTextractClientBuilder.defaultClient();

DetectDocumentTextRequest request = new DetectDocumentTextRequest()
        .withDocument(new Document()
                .withBytes(imageBytes));


DetectDocumentTextResult result = client.detectDocumentText(request);
```

Amazon S3 儲存貯體中的文件

Amazon Textract 可以分析存放在 Amazon S3 儲存貯體的文件影像。您指定儲存貯體和檔案名稱，方法是使用S3Object欄位Document輸入參數。以下範例示範 Amazon Textract 操作的輸入 JSON，該操作可處理存放在 Amazon S3 儲存貯體中的檔案。

```
{
    "Document": {
        "S3Object": {
            "Bucket": "bucket",
            "Name": "input.png"
        }
    }
}
```

以下範例示範如何使用存放在 Amazon S3 儲存貯體中的影像來呼叫 Amazon Amazon Textract 操作。

```
String document="input.png";
String bucket="bucket";

AmazonTextract client = AmazonTextractClientBuilder.defaultClient();

DetectDocumentTextRequest request = new DetectDocumentTextRequest()
        .withDocument(new Document()
                .withS3Object(new S3Object()
                        .withName(document)
                        .withBucket(bucket)));

DetectDocumentTextResult result = client.detectDocumentText(request);
```

回應

下列範例是來自呼叫的 JSON 回應DetectDocumentText。如需詳細資訊，請參閱 偵測文字。

```
{
{
  "DocumentMetadata": {
    "Pages": 1
  },
  "Blocks": [
    {
      "BlockType": "PAGE",
```

```
      "Geometry": {
        "BoundingBox": {
          "Width": 0.9995205998420715,
          "Height": 1.0,
          "Left": 0.0,
          "Top": 0.0
        },
        "Polygon": [
          {
            "X": 0.0,
            "Y": 0.0
          },
          {
            "X": 0.9995205998420715,
            "Y": 2.297314024515845E-16
          },
          {
            "X": 0.9995205998420715,
            "Y": 1.0
          },
          {
            "X": 0.0,
            "Y": 1.0
          }
        ]
      },
      "Id": "ca4b9171-7109-4adb-a811-e09bbe4834dd",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "26085884-d005-4144-b4c2-4d83dc50739b",
            "ee9d01bc-d91c-401d-8c0a-eec76f5f7862",
            "404bb3d3-d7ab-4008-a195-5dec87a08664",
            "8ae1b4ba-67c1-4486-bd20-54f461886ce9",
            "47aab5ab-be2c-4c73-97c7-d0a45454e843",
            "dd06bb49-6a56-4ea7-beec-a2aa09835c3c",
            "8837153d-81b8-4031-a49f-83a3d81803c2",
            "5dae3b74-9e95-4b62-99b7-93b88fe70648",
            "4508da80-64d8-42a8-8846-cfafe6eab10c",
            "e87be7a9-5519-42e1-b18e-ae10e2d3ed13",
            "f04bb223-d075-41c3-b328-7354611c826b",
            "a234f0e8-67de-46f4-a7c7-0bbe8d5159ce",
            "61b20e27-ff8a-450a-a8b1-bc0259f82fd6",
```

```
            "445f4fdd-c77b-4a7b-a2fc-6ca07cfe9ed7",
            "359f3870-7183-43f5-b638-970f5cefe4d5",
            "b9deea0a-244c-4d54-b774-cf03fbaaa8b1",
            "e2a43881-f620-44f2-b067-500ce7dc8d4d",
            "41756974-64ef-432d-b4b2-34702505975a",
            "93d96d32-8b4a-4a98-9578-8b4df4f227a6",
            "bc907357-63d6-43c0-ab87-80d7e76d377e",
            "2d727ca7-3acb-4bb9-a564-5885c90e9325",
            "f32a5989-cbfb-41e6-b0fc-ce1c77c014bd",
            "e0ba06d0-dbb6-4962-8047-8cac3adfe45a",
            "b6ed204d-ae01-4b75-bb91-c85d4147a37e",
            "ac4b9ee0-c9b2-4239-a741-5753e5282033",
            "ebc18885-48d7-45b8-90e3-d172b4357802",
            "babf6360-789e-49c1-9c78-0784acc14a0c"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.93761444091797,
      "Text": "Employment Application",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.3391372561454773,
          "Height": 0.06906412541866302,
          "Left": 0.29548385739326477,
          "Top": 0.027493247762322426
        },
        "Polygon": [
          {
            "X": 0.29548385739326477,
            "Y": 0.027493247762322426
          },
          {
            "X": 0.6346210837364197,
            "Y": 0.027493247762322426
          },
          {
            "X": 0.6346210837364197,
            "Y": 0.0965573713183403
          },
          {
            "X": 0.29548385739326477,
```

```
            "Y": 0.0965573713183403
          }
        ]
      },
      "Id": "26085884-d005-4144-b4c2-4d83dc50739b",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "ed48dacc-d089-498f-8e93-1cee1e5f39f3",
            "ac7370f3-cbb7-4cd9-a8f9-bdcb2252caaf"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.91246795654297,
      "Text": "Application Information",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.19878505170345306,
          "Height": 0.03754019737243652,
          "Left": 0.03988289833068848,
          "Top": 0.14050349593162537
        },
        "Polygon": [
          {
            "X": 0.03988289833068848,
            "Y": 0.14050349593162537
          },
          {
            "X": 0.23866795003414154,
            "Y": 0.14050349593162537
          },
          {
            "X": 0.23866795003414154,
            "Y": 0.1780436933040619
          },
          {
            "X": 0.03988289833068848,
            "Y": 0.1780436933040619
          }
        ]
```

```
      },
      "Id": "ee9d01bc-d91c-401d-8c0a-eec76f5f7862",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "efe3fc6d-becb-4520-80ee-49a329386aee",
            "c2260852-6cfd-4a71-9fc6-62b2f9b02355"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.88693237304688,
      "Text": "Full Name: Jane Doe",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.16733919084072113,
          "Height": 0.031106337904930115,
          "Left": 0.03899926319718361,
          "Top": 0.21361036598682404
        },
        "Polygon": [
          {
            "X": 0.03899926319718361,
            "Y": 0.21361036598682404
          },
          {
            "X": 0.20633845031261444,
            "Y": 0.21361036598682404
          },
          {
            "X": 0.20633845031261444,
            "Y": 0.24471670389175415
          },
          {
            "X": 0.03899926319718361,
            "Y": 0.24471670389175415
          }
        ]
      },
      "Id": "404bb3d3-d7ab-4008-a195-5dec87a08664",
      "Relationships": [
```

```
        {
          "Type": "CHILD",
          "Ids": [
            "e94eb587-9545-4215-b0fc-8e8cb1172958",
            "090aeba5-8428-4b7a-a54b-7a95a774120e",
            "64ff0abb-736b-4a6b-aa8d-ad2c0086ae1d",
            "565ffc30-89d6-4295-b8c6-d22b4ed76584"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.9206314086914,
      "Text": "Phone Number: 555-0100",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.3115004599094391,
          "Height": 0.047169625759124756,
          "Left": 0.03604753687977791,
          "Top": 0.2812676727771759
        },
        "Polygon": [
          {
            "X": 0.03604753687977791,
            "Y": 0.2812676727771759
          },
          {
            "X": 0.3475480079650879,
            "Y": 0.2812676727771759
          },
          {
            "X": 0.3475480079650879,
            "Y": 0.32843729853630066
          },
          {
            "X": 0.03604753687977791,
            "Y": 0.32843729853630066
          }
        ]
      },
      "Id": "8ae1b4ba-67c1-4486-bd20-54f461886ce9",
      "Relationships": [
        {
```

```
          "Type": "CHILD",
          "Ids": [
            "d782f847-225b-4a1b-b52d-f252f8221b1f",
            "fa69c5cd-c80d-4fac-81df-569edae8d259",
            "d4bbc0f1-ae02-41cf-a26f-8a1e899968cc"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.48902893066406,
      "Text": "Home Address: 123 Any Street, Any Town. USA",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.7431139945983887,
          "Height": 0.09577702730894089,
          "Left": 0.03359385207295418,
          "Top": 0.3258342146873474
        },
        "Polygon": [
          {
            "X": 0.03359385207295418,
            "Y": 0.3258342146873474
          },
          {
            "X": 0.7767078280448914,
            "Y": 0.3258342146873474
          },
          {
            "X": 0.7767078280448914,
            "Y": 0.4216112196445465
          },
          {
            "X": 0.03359385207295418,
            "Y": 0.4216112196445465
          }
        ]
      },
      "Id": "47aab5ab-be2c-4c73-97c7-d0a45454e843",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
```

```
            "acfbed90-4a00-42c6-8a90-d0a0756eea36",
            "046c8a40-bb0e-4718-9c71-954d3630e1dd",
            "82b838bc-4591-4287-8dea-60c94a4925e4",
            "5cdcde7a-f5a6-4231-a941-b6396e42e7ba",
            "beafd497-185f-487e-b070-db4df5803e94",
            "ef1b77fb-8ba6-41fe-ba53-dce039af22ed",
            "7b555310-e7f8-4cd2-bb3d-dcec37f3d90e",
            "b479c24d-448d-40ef-9ed5-36a6ef08e5c7"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.89382934570312,
      "Text": "Mailing Address: same as above",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.26575741171836853,
          "Height": 0.039571404457092285,
          "Left": 0.03068041242659092,
          "Top": 0.43351811170578003
        },
        "Polygon": [
          {
            "X": 0.03068041242659092,
            "Y": 0.43351811170578003
          },
          {
            "X": 0.2964377999305725,
            "Y": 0.43351811170578003
          },
          {
            "X": 0.2964377999305725,
            "Y": 0.4730895161628723
          },
          {
            "X": 0.03068041242659092,
            "Y": 0.4730895161628723
          }
        ]
      },
      "Id": "dd06bb49-6a56-4ea7-beec-a2aa09835c3c",
      "Relationships": [
```

```
        {
          "Type": "CHILD",
          "Ids": [
            "d7261cdc-6ac5-4711-903c-4598fe94952d",
            "287f80c3-6db2-4dd7-90ec-5f017c80aa31",
            "ce31c3ad-b51e-4068-be64-5fc9794bc1bc",
            "e96eb92c-6774-4d6f-8f4a-68a7618d4c66",
            "88b85c05-427a-4d4f-8cc4-3667234e8364"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 94.67343139648438,
      "Text": "Previous Employment History",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.3309842050075531,
          "Height": 0.051920413970947266,
          "Left": 0.3194798231124878,
          "Top": 0.5172380208969116
        },
        "Polygon": [
          {
            "X": 0.3194798231124878,
            "Y": 0.5172380208969116
          },
          {
            "X": 0.6504639983177185,
            "Y": 0.5172380208969116
          },
          {
            "X": 0.6504639983177185,
            "Y": 0.5691584348678589
          },
          {
            "X": 0.3194798231124878,
            "Y": 0.5691584348678589
          }
        ]
      },
      "Id": "8837153d-81b8-4031-a49f-83a3d81803c2",
      "Relationships": [
```

```
        {
          "Type": "CHILD",
          "Ids": [
            "8b324501-bf38-4ce9-9777-6514b7ade760",
            "b0cea99a-5045-464d-ac8a-a63ab0470995",
            "b92a6ee5-ca59-44dc-9c47-534c133b11e7"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.66949462890625,
      "Text": "Start Date",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08310240507125854,
          "Height": 0.030944595113396645,
          "Left": 0.034429505467414856,
          "Top": 0.6123942136764526
        },
        "Polygon": [
          {
            "X": 0.034429505467414856,
            "Y": 0.6123942136764526
          },
          {
            "X": 0.1175319030880928,
            "Y": 0.6123942136764526
          },
          {
            "X": 0.1175319030880928,
            "Y": 0.6433387994766235
          },
          {
            "X": 0.034429505467414856,
            "Y": 0.6433387994766235
          }
        ]
      },
      "Id": "5dae3b74-9e95-4b62-99b7-93b88fe70648",
      "Relationships": [
        {
          "Type": "CHILD",
```

```
          "Ids": [
            "ffe8b8e0-df59-4ac5-9aba-6b54b7c51b45",
            "91e582cd-9871-4e9c-93cc-848baa426338"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.86717224121094,
      "Text": "End Date",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.07581500709056854,
          "Height": 0.03223184868693352,
          "Left": 0.14846202731132507,
          "Top": 0.6120467782020569
        },
        "Polygon": [
          {
            "X": 0.14846202731132507,
            "Y": 0.6120467782020569
          },
          {
            "X": 0.22427703440189362,
            "Y": 0.6120467782020569
          },
          {
            "X": 0.22427703440189362,
            "Y": 0.6442786455154419
          },
          {
            "X": 0.14846202731132507,
            "Y": 0.6442786455154419
          }
        ]
      },
      "Id": "4508da80-64d8-42a8-8846-cfafe6eab10c",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "7c97b56b-699f-49b0-93f4-98e6d90b107c",
            "7af04e27-0c15-447e-a569-b30edb99a133"
```

```
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.9539794921875,
      "Text": "Employer Name",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.1347292959690094,
          "Height": 0.0392492413520813,
          "Left": 0.2647075653076172,
          "Top": 0.6140711903572083
        },
        "Polygon": [
          {
            "X": 0.2647075653076172,
            "Y": 0.6140711903572083
          },
          {
            "X": 0.3994368314743042,
            "Y": 0.6140711903572083
          },
          {
            "X": 0.3994368314743042,
            "Y": 0.6533204317092896
          },
          {
            "X": 0.2647075653076172,
            "Y": 0.6533204317092896
          }
        ]
      },
      "Id": "e87be7a9-5519-42e1-b18e-ae10e2d3ed13",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "a9bfeb55-75cd-47cd-b953-728e602a3564",
            "9f0f9c06-d02c-4b07-bb39-7ade70be2c1b"
          ]
        }
      ]
```

```
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.35584259033203,
      "Text": "Position Held",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.11393272876739502,
          "Height": 0.03415105864405632,
          "Left": 0.49973347783088684,
          "Top": 0.614840030670166
        },
        "Polygon": [
          {
            "X": 0.49973347783088684,
            "Y": 0.614840030670166
          },
          {
            "X": 0.6136661767959595,
            "Y": 0.614840030670166
          },
          {
            "X": 0.6136661767959595,
            "Y": 0.6489911079406738
          },
          {
            "X": 0.49973347783088684,
            "Y": 0.6489911079406738
          }
        ]
      },
      "Id": "f04bb223-d075-41c3-b328-7354611c826b",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "6d5edf02-845c-40e0-9514-e56d0d652ae0",
            "3297ab59-b237-45fb-ae60-a108f0c95ac2"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
```

```
      "Confidence": 99.9817886352539,
      "Text": "Reason for leaving",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.16511960327625275,
          "Height": 0.04062700271606445,
          "Left": 0.7430596351623535,
          "Top": 0.6116235852241516
        },
        "Polygon": [
          {
            "X": 0.7430596351623535,
            "Y": 0.6116235852241516
          },
          {
            "X": 0.9081792235374451,
            "Y": 0.6116235852241516
          },
          {
            "X": 0.9081792235374451,
            "Y": 0.6522505879402161
          },
          {
            "X": 0.7430596351623535,
            "Y": 0.6522505879402161
          }
        ]
      },
      "Id": "a234f0e8-67de-46f4-a7c7-0bbe8d5159ce",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "f4b8cf26-d2da-4a76-8345-69562de3cc11",
            "386d4a63-1194-4c0e-a18d-4d074a0b1f93",
            "a8622541-1896-4d54-8d10-7da2c800ec5c"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.77413177490234,
      "Text": "1/15/2009",
```

```
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08799663186073303,
          "Height": 0.03832906484603882,
          "Left": 0.03175082430243492,
          "Top": 0.691371738910675
        },
        "Polygon": [
          {
            "X": 0.03175082430243492,
            "Y": 0.691371738910675
          },
          {
            "X": 0.11974745243787766,
            "Y": 0.691371738910675
          },
          {
            "X": 0.11974745243787766,
            "Y": 0.7297008037567139
          },
          {
            "X": 0.03175082430243492,
            "Y": 0.7297008037567139
          }
        ]
      },
      "Id": "61b20e27-ff8a-450a-a8b1-bc0259f82fd6",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "da7a6482-0964-49a4-bc7d-56942ff3b4e1"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.72286224365234,
      "Text": "6/30/2011",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08843101561069489,
          "Height": 0.03991425037384033,
```

```
          "Left": 0.14642837643623352,
          "Top": 0.6919752955436707
        },
        "Polygon": [
          {
            "X": 0.14642837643623352,
            "Y": 0.6919752955436707
          },
          {
            "X": 0.2348593920469284,
            "Y": 0.6919752955436707
          },
          {
            "X": 0.2348593920469284,
            "Y": 0.731889545917511
          },
          {
            "X": 0.14642837643623352,
            "Y": 0.731889545917511
          }
        ]
      },
      "Id": "445f4fdd-c77b-4a7b-a2fc-6ca07cfe9ed7",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "5a8da66a-ecce-4ee9-a765-a46d6cdc6cde"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.86936950683594,
      "Text": "Any Company",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.11800950765609741,
          "Height": 0.03943679481744766,
          "Left": 0.2626699209213257,
          "Top": 0.6972727179527283
        },
        "Polygon": [
```

```
          {
            "X": 0.2626699209213257,
            "Y": 0.6972727179527283
          },
          {
            "X": 0.3806794285774231,
            "Y": 0.6972727179527283
          },
          {
            "X": 0.3806794285774231,
            "Y": 0.736709475517273
          },
          {
            "X": 0.2626699209213257,
            "Y": 0.736709475517273
          }
        ]
      },
      "Id": "359f3870-7183-43f5-b638-970f5cefe4d5",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "77749c2b-aa7f-450e-8dd2-62bcaf253ba2",
            "713bad19-158d-4e3e-b01f-f5707ddb04e5"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.582275390625,
      "Text": "Assistant baker",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.13280922174453735,
          "Height": 0.032666124403476715,
          "Left": 0.49814170598983765,
          "Top": 0.699238657951355
        },
        "Polygon": [
          {
            "X": 0.49814170598983765,
            "Y": 0.699238657951355
```

```
          },
          {
            "X": 0.630950927734375,
            "Y": 0.699238657951355
          },
          {
            "X": 0.630950927734375,
            "Y": 0.7319048047065735
          },
          {
            "X": 0.49814170598983765,
            "Y": 0.7319048047065735
          }
        ]
      },
      "Id": "b9deea0a-244c-4d54-b774-cf03fbaaa8b1",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "989944f9-f684-4714-87d8-9ad9a321d65c",
            "ae82e2aa-1601-4e0c-8340-1db7ad0c9a31"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.96180725097656,
      "Text": "relocated",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08668994903564453,
          "Height": 0.033302485942840576,
          "Left": 0.7426905632019043,
          "Top": 0.6974037289619446
        },
        "Polygon": [
          {
            "X": 0.7426905632019043,
            "Y": 0.6974037289619446
          },
          {
            "X": 0.8293805122375488,
```

```
            "Y": 0.6974037289619446
          },
          {
            "X": 0.8293805122375488,
            "Y": 0.7307062149047852
          },
          {
            "X": 0.7426905632019043,
            "Y": 0.7307062149047852
          }
        ]
      },
      "Id": "e2a43881-f620-44f2-b067-500ce7dc8d4d",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "a9cf9a8c-fdaa-413e-9346-5a28a98aebdb"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.98190307617188,
      "Text": "7/1/2011",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.09747002273797989,
          "Height": 0.07067441940307617,
          "Left": 0.028500309213995934,
          "Top": 0.7745237946510315
        },
        "Polygon": [
          {
            "X": 0.028500309213995934,
            "Y": 0.7745237946510315
          },
          {
            "X": 0.12597033381462097,
            "Y": 0.7745237946510315
          },
          {
            "X": 0.12597033381462097,
```

```
            "Y": 0.8451982140541077
          },
          {
            "X": 0.028500309213995934,
            "Y": 0.8451982140541077
          }
        ]
      },
      "Id": "41756974-64ef-432d-b4b2-34702505975a",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "0f711065-1872-442a-ba6d-8fababaa452a"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.98418426513672,
      "Text": "8/10/2013",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.10664612054824829,
          "Height": 0.06439518928527832,
          "Left": 0.14159755408763885,
          "Top": 0.7791688442230225
        },
        "Polygon": [
          {
            "X": 0.14159755408763885,
            "Y": 0.7791688442230225
          },
          {
            "X": 0.24824367463588715,
            "Y": 0.7791688442230225
          },
          {
            "X": 0.24824367463588715,
            "Y": 0.8435640335083008
          },
          {
            "X": 0.14159755408763885,
```

```
            "Y": 0.8435640335083008
          }
        ]
      },
      "Id": "93d96d32-8b4a-4a98-9578-8b4df4f227a6",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "a92d8eef-db28-45ba-801a-5da0f589d277"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.98075866699219,
      "Text": "Example Corp.",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.2114926278591156,
          "Height": 0.058415766805410385,
          "Left": 0.26764172315597534,
          "Top": 0.794414758682251
        },
        "Polygon": [
          {
            "X": 0.26764172315597534,
            "Y": 0.794414758682251
          },
          {
            "X": 0.47913435101509094,
            "Y": 0.794414758682251
          },
          {
            "X": 0.47913435101509094,
            "Y": 0.8528305292129517
          },
          {
            "X": 0.26764172315597534,
            "Y": 0.8528305292129517
          }
        ]
      },
```

```
      "Id": "bc907357-63d6-43c0-ab87-80d7e76d377e",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "d6962efb-34ab-4ffb-9f2f-5f263e813558",
            "1876c8ea-d3e8-4c39-870e-47512b3b5080"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.91166687011719,
      "Text": "Baker",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.09931200742721558,
          "Height": 0.06008726358413696,
          "Left": 0.5098910331726074,
          "Top": 0.787897527217865
        },
        "Polygon": [
          {
            "X": 0.5098910331726074,
            "Y": 0.787897527217865
          },
          {
            "X": 0.609203040599823,
            "Y": 0.787897527217865
          },
          {
            "X": 0.609203040599823,
            "Y": 0.847984790802002
          },
          {
            "X": 0.5098910331726074,
            "Y": 0.847984790802002
          }
        ]
      },
      "Id": "2d727ca7-3acb-4bb9-a564-5885c90e9325",
      "Relationships": [
        {
```

```
          "Type": "CHILD",
          "Ids": [
            "00adeaef-ed57-44eb-b8a9-503575236d62"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.93852233886719,
      "Text": "better opp.",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.18919607996940613,
          "Height": 0.06994765996932983,
          "Left": 0.7428008317947388,
          "Top": 0.7928366661071777
        },
        "Polygon": [
          {
            "X": 0.7428008317947388,
            "Y": 0.7928366661071777
          },
          {
            "X": 0.9319968819618225,
            "Y": 0.7928366661071777
          },
          {
            "X": 0.9319968819618225,
            "Y": 0.8627843260765076
          },
          {
            "X": 0.7428008317947388,
            "Y": 0.8627843260765076
          }
        ]
      },
      "Id": "f32a5989-cbfb-41e6-b0fc-ce1c77c014bd",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "c0fc9a58-7a4b-4f69-bafd-2cff32be2665",
            "bf6dc8ee-2fb3-4b6c-aee4-31e96912a2d8"
```

```
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.92573547363281,
      "Text": "8/15/2013",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.10257463902235031,
          "Height": 0.05412459373474121,
          "Left": 0.027909137308597565,
          "Top": 0.8608770370483398
        },
        "Polygon": [
          {
            "X": 0.027909137308597565,
            "Y": 0.8608770370483398
          },
          {
            "X": 0.13048377633094788,
            "Y": 0.8608770370483398
          },
          {
            "X": 0.13048377633094788,
            "Y": 0.915001630783081
          },
          {
            "X": 0.027909137308597565,
            "Y": 0.915001630783081
          }
        ]
      },
      "Id": "e0ba06d0-dbb6-4962-8047-8cac3adfe45a",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "5384f860-f857-4a94-9438-9dfa20eed1c6"
          ]
        }
      ]
    },
```

```
    {
      "BlockType": "LINE",
      "Confidence": 99.99625396728516,
      "Text": "Present",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.09982697665691376,
          "Height": 0.06888341903686523,
          "Left": 0.1420602649450302,
          "Top": 0.8511748909950256
        },
        "Polygon": [
          {
            "X": 0.1420602649450302,
            "Y": 0.8511748909950256
          },
          {
            "X": 0.24188724160194397,
            "Y": 0.8511748909950256
          },
          {
            "X": 0.24188724160194397,
            "Y": 0.9200583100318909
          },
          {
            "X": 0.1420602649450302,
            "Y": 0.9200583100318909
          }
        ]
      },
      "Id": "b6ed204d-ae01-4b75-bb91-c85d4147a37e",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "0bb96ed6-b2e6-4da4-90b3-b85561bbd89d"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.9826431274414,
      "Text": "AnyCompany",
```

```
      "Geometry": {
        "BoundingBox": {
          "Width": 0.18611276149749756,
          "Height": 0.08581399917602539,
          "Left": 0.2615866959095001,
          "Top": 0.869536280632019
        },
        "Polygon": [
          {
            "X": 0.2615866959095001,
            "Y": 0.869536280632019
          },
          {
            "X": 0.4476994574069977,
            "Y": 0.869536280632019
          },
          {
            "X": 0.4476994574069977,
            "Y": 0.9553502798080444
          },
          {
            "X": 0.2615866959095001,
            "Y": 0.9553502798080444
          }
        ]
      },
      "Id": "ac4b9ee0-c9b2-4239-a741-5753e5282033",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "25343360-d906-440a-88b7-92eb89e95949"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.99549102783203,
      "Text": "head baker",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.1937451809644699,
          "Height": 0.056156039237976074,
```

```
          "Left": 0.49359121918678284,
          "Top": 0.8702592849731445
        },
        "Polygon": [
          {
            "X": 0.49359121918678284,
            "Y": 0.8702592849731445
          },
          {
            "X": 0.6873363852500916,
            "Y": 0.8702592849731445
          },
          {
            "X": 0.6873363852500916,
            "Y": 0.9264153242111206
          },
          {
            "X": 0.49359121918678284,
            "Y": 0.9264153242111206
          }
        ]
      },
      "Id": "ebc18885-48d7-45b8-90e3-d172b4357802",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "0ef3c194-8322-4575-94f1-82819ee57e3a",
            "d296acd9-3e9a-4985-95f8-f863614f2c46"
          ]
        }
      ]
    },
    {
      "BlockType": "LINE",
      "Confidence": 99.98360443115234,
      "Text": "N/A, current",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.22544169425964355,
          "Height": 0.06588292121887207,
          "Left": 0.7411766648292542,
          "Top": 0.8722732067108154
        },
```

```
        "Polygon": [
          {
            "X": 0.7411766648292542,
            "Y": 0.8722732067108154
          },
          {
            "X": 0.9666183590888977,
            "Y": 0.8722732067108154
          },
          {
            "X": 0.9666183590888977,
            "Y": 0.9381561279296875
          },
          {
            "X": 0.7411766648292542,
            "Y": 0.9381561279296875
          }
        ]
      },
      "Id": "babf6360-789e-49c1-9c78-0784acc14a0c",
      "Relationships": [
        {
          "Type": "CHILD",
          "Ids": [
            "195cfb5b-ae06-4203-8520-4e4b0a73b5ce",
            "549ef3f9-3a13-4b77-bc25-fb2e0996839a"
          ]
        }
      ]
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.94815826416016,
      "Text": "Employment",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.17462396621704102,
          "Height": 0.06266549974679947,
          "Left": 0.29548385739326477,
          "Top": 0.03389188274741173
        },
        "Polygon": [
          {
```

```
            "X": 0.29548385739326477,
            "Y": 0.03389188274741173
          },
          {
            "X": 0.4701078236103058,
            "Y": 0.03389188274741173
          },
          {
            "X": 0.4701078236103058,
            "Y": 0.0965573862195015
          },
          {
            "X": 0.29548385739326477,
            "Y": 0.0965573862195015
          }
        ]
      },
      "Id": "ed48dacc-d089-498f-8e93-1cee1e5f39f3"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.92706298828125,
      "Text": "Application",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.15933875739574432,
          "Height": 0.062391020357608795,
          "Left": 0.47528234124183655,
          "Top": 0.027493247762322426
        },
        "Polygon": [
          {
            "X": 0.47528234124183655,
            "Y": 0.027493247762322426
          },
          {
            "X": 0.6346211433410645,
            "Y": 0.027493247762322426
          },
          {
            "X": 0.6346211433410645,
            "Y": 0.08988427370786667
          },
```

```
          {
            "X": 0.47528234124183655,
            "Y": 0.08988427370786667
          }
        ]
      },
      "Id": "ac7370f3-cbb7-4cd9-a8f9-bdcb2252caaf"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.9821548461914,
      "Text": "Application",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.09610454738140106,
          "Height": 0.03656719997525215,
          "Left": 0.03988289833068848,
          "Top": 0.14147649705410004
        },
        "Polygon": [
          {
            "X": 0.03988289833068848,
            "Y": 0.14147649705410004
          },
          {
            "X": 0.13598744571208954,
            "Y": 0.14147649705410004
          },
          {
            "X": 0.13598744571208954,
            "Y": 0.1780436933040619
          },
          {
            "X": 0.03988289833068848,
            "Y": 0.1780436933040619
          }
        ]
      },
      "Id": "efe3fc6d-becb-4520-80ee-49a329386aee"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.84278106689453,
```

```
      "Text": "Information",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.10029315203428268,
          "Height": 0.03209415823221207,
          "Left": 0.13837480545043945,
          "Top": 0.14050349593162537
        },
        "Polygon": [
          {
            "X": 0.13837480545043945,
            "Y": 0.14050349593162537
          },
          {
            "X": 0.23866795003414154,
            "Y": 0.14050349593162537
          },
          {
            "X": 0.23866795003414154,
            "Y": 0.17259766161441803
          },
          {
            "X": 0.13837480545043945,
            "Y": 0.17259766161441803
          }
        ]
      },
      "Id": "c2260852-6cfd-4a71-9fc6-62b2f9b02355"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.83993530273438,
      "Text": "Full",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.03039788082242012,
          "Height": 0.031106330454349518,
          "Left": 0.03899926319718361,
          "Top": 0.21361036598682404
        },
        "Polygon": [
          {
```

```
            "X": 0.03899926319718361,
            "Y": 0.21361036598682404
          },
          {
            "X": 0.06939714401960373,
            "Y": 0.21361036598682404
          },
          {
            "X": 0.06939714401960373,
            "Y": 0.24471670389175415
          },
          {
            "X": 0.03899926319718361,
            "Y": 0.24471670389175415
          }
        ]
      },
      "Id": "e94eb587-9545-4215-b0fc-8e8cb1172958"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.93611907958984,
      "Text": "Name:",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.05555811896920204,
          "Height": 0.030184319242835045,
          "Left": 0.07123806327581406,
          "Top": 0.2137702852487564
        },
        "Polygon": [
          {
            "X": 0.07123806327581406,
            "Y": 0.2137702852487564
          },
          {
            "X": 0.1267961859703064,
            "Y": 0.2137702852487564
          },
          {
            "X": 0.1267961859703064,
            "Y": 0.2439546138048172
          },
```

```
        {
          "X": 0.07123806327581406,
          "Y": 0.2439546138048172
        }
      ]
    },
    "Id": "090aeba5-8428-4b7a-a54b-7a95a774120e"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.91043853759766,
    "Text": "Jane",
    "TextType": "PRINTED",
    "Geometry": {
      "BoundingBox": {
        "Width": 0.03905024006962776,
        "Height": 0.02941947989165783,
        "Left": 0.12933772802352905,
        "Top": 0.214289128780365
      },
      "Polygon": [
        {
          "X": 0.12933772802352905,
          "Y": 0.214289128780365
        },
        {
          "X": 0.16838796436786652,
          "Y": 0.214289128780365
        },
        {
          "X": 0.16838796436786652,
          "Y": 0.24370861053466797
        },
        {
          "X": 0.12933772802352905,
          "Y": 0.24370861053466797
        }
      ]
    },
    "Id": "64ff0abb-736b-4a6b-aa8d-ad2c0086ae1d"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.86123657226562,
```

```
      "Text": "Doe",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.035229459404945374,
          "Height": 0.030427640303969383,
          "Left": 0.17110899090766907,
          "Top": 0.21377210319042206
        },
        "Polygon": [
          {
            "X": 0.17110899090766907,
            "Y": 0.21377210319042206
          },
          {
            "X": 0.20633845031261444,
            "Y": 0.21377210319042206
          },
          {
            "X": 0.20633845031261444,
            "Y": 0.244199737906456
          },
          {
            "X": 0.17110899090766907,
            "Y": 0.244199737906456
          }
        ]
      },
      "Id": "565ffc30-89d6-4295-b8c6-d22b4ed76584"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.92633056640625,
      "Text": "Phone",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.052783288061618805,
          "Height": 0.03104414977133274,
          "Left": 0.03604753687977791,
          "Top": 0.28701552748680115
        },
        "Polygon": [
          {
```

```
            "X": 0.03604753687977791,
            "Y": 0.28701552748680115
          },
          {
            "X": 0.08883082121610641,
            "Y": 0.28701552748680115
          },
          {
            "X": 0.08883082121610641,
            "Y": 0.31805968284606934
          },
          {
            "X": 0.03604753687977791,
            "Y": 0.31805968284606934
          }
        ]
      },
      "Id": "d782f847-225b-4a1b-b52d-f252f8221b1f"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.86275482177734,
      "Text": "Number:",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.07424934208393097,
          "Height": 0.030300479382276535,
          "Left": 0.0915418416261673,
          "Top": 0.28639692068099976
        },
        "Polygon": [
          {
            "X": 0.0915418416261673,
            "Y": 0.28639692068099976
          },
          {
            "X": 0.16579118371009827,
            "Y": 0.28639692068099976
          },
          {
            "X": 0.16579118371009827,
            "Y": 0.3166973888874054
          },
```

```
        {
          "X": 0.0915418416261673,
          "Y": 0.3166973888874054
        }
      ]
    },
    "Id": "fa69c5cd-c80d-4fac-81df-569edae8d259"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.97282409667969,
    "Text": "555-0100",
    "TextType": "HANDWRITING",
    "Geometry": {
      "BoundingBox": {
        "Width": 0.17021971940994263,
        "Height": 0.047169629484415054,
        "Left": 0.17732827365398407,
        "Top": 0.2812676727771759
      },
      "Polygon": [
        {
          "X": 0.17732827365398407,
          "Y": 0.2812676727771759
        },
        {
          "X": 0.3475480079650879,
          "Y": 0.2812676727771759
        },
        {
          "X": 0.3475480079650879,
          "Y": 0.32843729853630066
        },
        {
          "X": 0.17732827365398407,
          "Y": 0.32843729853630066
        }
      ]
    },
    "Id": "d4bbc0f1-ae02-41cf-a26f-8a1e899968cc"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.66238403320312,
```

```
      "Text": "Home",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.049357783049345016,
          "Height": 0.03134990110993385,
          "Left": 0.03359385207295418,
          "Top": 0.36172014474868774
        },
        "Polygon": [
          {
            "X": 0.03359385207295418,
            "Y": 0.36172014474868774
          },
          {
            "X": 0.0829516351222992,
            "Y": 0.36172014474868774
          },
          {
            "X": 0.0829516351222992,
            "Y": 0.3930700421333313
          },
          {
            "X": 0.03359385207295418,
            "Y": 0.3930700421333313
          }
        ]
      },
      "Id": "acfbed90-4a00-42c6-8a90-d0a0756eea36"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.6871109008789,
      "Text": "Address:",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.07411003112792969,
          "Height": 0.0314042791724205,
          "Left": 0.08516156673431396,
          "Top": 0.3600046932697296
        },
        "Polygon": [
          {
```

```
            "X": 0.08516156673431396,
            "Y": 0.3600046932697296
          },
          {
            "X": 0.15927159786224365,
            "Y": 0.3600046932697296
          },
          {
            "X": 0.15927159786224365,
            "Y": 0.3914089798927307
          },
          {
            "X": 0.08516156673431396,
            "Y": 0.3914089798927307
          }
        ]
      },
      "Id": "046c8a40-bb0e-4718-9c71-954d3630e1dd"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.93781280517578,
      "Text": "123",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.05761868134140968,
          "Height": 0.05008566007018089,
          "Left": 0.1750781387090683,
          "Top": 0.35484206676483154
        },
        "Polygon": [
          {
            "X": 0.1750781387090683,
            "Y": 0.35484206676483154
          },
          {
            "X": 0.23269681632518768,
            "Y": 0.35484206676483154
          },
          {
            "X": 0.23269681632518768,
            "Y": 0.40492773056030273
          },
```

```
          {
            "X": 0.1750781387090683,
            "Y": 0.40492773056030273
          }
        ]
      },
      "Id": "82b838bc-4591-4287-8dea-60c94a4925e4"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.96530151367188,
      "Text": "Any",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.06814215332269669,
          "Height": 0.06354366987943649,
          "Left": 0.2550157308578491,
          "Top": 0.35471394658088684
        },
        "Polygon": [
          {
            "X": 0.2550157308578491,
            "Y": 0.35471394658088684
          },
          {
            "X": 0.3231579065322876,
            "Y": 0.35471394658088684
          },
          {
            "X": 0.3231579065322876,
            "Y": 0.41825762391090393
          },
          {
            "X": 0.2550157308578491,
            "Y": 0.41825762391090393
          }
        ]
      },
      "Id": "5cdcde7a-f5a6-4231-a941-b6396e42e7ba"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.87527465820312,
```

```
      "Text": "Street,",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.12156613171100616,
          "Height": 0.05449587106704712,
          "Left": 0.3357025980949402,
          "Top": 0.3550415635108948
        },
        "Polygon": [
          {
            "X": 0.3357025980949402,
            "Y": 0.3550415635108948
          },
          {
            "X": 0.45726871490478516,
            "Y": 0.3550415635108948
          },
          {
            "X": 0.45726871490478516,
            "Y": 0.4095374345779419
          },
          {
            "X": 0.3357025980949402,
            "Y": 0.4095374345779419
          }
        ]
      },
      "Id": "beafd497-185f-487e-b070-db4df5803e94"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.99514770507812,
      "Text": "Any",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.07748188823461533,
          "Height": 0.07339789718389511,
          "Left": 0.47723668813705444,
          "Top": 0.3482133150100708
        },
        "Polygon": [
          {
```

```
            "X": 0.47723668813705444,
            "Y": 0.3482133150100708
          },
          {
            "X": 0.554718554019928,
            "Y": 0.3482133150100708
          },
          {
            "X": 0.554718554019928,
            "Y": 0.4216112196445465
          },
          {
            "X": 0.47723668813705444,
            "Y": 0.4216112196445465
          }
        ]
      },
      "Id": "ef1b77fb-8ba6-41fe-ba53-dce039af22ed"
    },
    {
      "BlockType": "WORD",
      "Confidence": 96.80656433105469,
      "Text": "Town.",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.11213835328817368,
          "Height": 0.057233039289712906,
          "Left": 0.5563329458236694,
          "Top": 0.3331930637359619
        },
        "Polygon": [
          {
            "X": 0.5563329458236694,
            "Y": 0.3331930637359619
          },
          {
            "X": 0.6684713363647461,
            "Y": 0.3331930637359619
          },
          {
            "X": 0.6684713363647461,
            "Y": 0.3904260993003845
          },
```

```
          {
            "X": 0.5563329458236694,
            "Y": 0.3904260993003845
          }
        ]
      },
      "Id": "7b555310-e7f8-4cd2-bb3d-dcec37f3d90e"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.98260498046875,
      "Text": "USA",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08771833777427673,
          "Height": 0.05706935003399849,
          "Left": 0.6889894604682922,
          "Top": 0.3258342146873474
        },
        "Polygon": [
          {
            "X": 0.6889894604682922,
            "Y": 0.3258342146873474
          },
          {
            "X": 0.7767078280448914,
            "Y": 0.3258342146873474
          },
          {
            "X": 0.7767078280448914,
            "Y": 0.3829035460948944
          },
          {
            "X": 0.6889894604682922,
            "Y": 0.3829035460948944
          }
        ]
      },
      "Id": "b479c24d-448d-40ef-9ed5-36a6ef08e5c7"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.9583969116211,
```

```
      "Text": "Mailing",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.06291338801383972,
          "Height": 0.03957144916057587,
          "Left": 0.03068041242659092,
          "Top": 0.43351811170578003
        },
        "Polygon": [
          {
            "X": 0.03068041242659092,
            "Y": 0.43351811170578003
          },
          {
            "X": 0.09359379857778549,
            "Y": 0.43351811170578003
          },
          {
            "X": 0.09359379857778549,
            "Y": 0.4730895459651947
          },
          {
            "X": 0.03068041242659092,
            "Y": 0.4730895459651947
          }
        ]
      },
      "Id": "d7261cdc-6ac5-4711-903c-4598fe94952d"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.87476348876953,
      "Text": "Address:",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.07364854216575623,
          "Height": 0.03147412836551666,
          "Left": 0.0954652726650238,
          "Top": 0.43450701236724854
        },
        "Polygon": [
          {
```

```
            "X": 0.0954652726650238,
            "Y": 0.43450701236724854
          },
          {
            "X": 0.16911381483078003,
            "Y": 0.43450701236724854
          },
          {
            "X": 0.16911381483078003,
            "Y": 0.465981125831604
          },
          {
            "X": 0.0954652726650238,
            "Y": 0.465981125831604
          }
        ]
      },
      "Id": "287f80c3-6db2-4dd7-90ec-5f017c80aa31"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.94071960449219,
      "Text": "same",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.04640670120716095,
          "Height": 0.026415130123496056,
          "Left": 0.17156922817230225,
          "Top": 0.44010937213897705
        },
        "Polygon": [
          {
            "X": 0.17156922817230225,
            "Y": 0.44010937213897705
          },
          {
            "X": 0.2179759293794632,
            "Y": 0.44010937213897705
          },
          {
            "X": 0.2179759293794632,
            "Y": 0.46652451157569885
          },
```

```
        {
          "X": 0.17156922817230225,
          "Y": 0.46652451157569885
        }
      ]
    },
    "Id": "ce31c3ad-b51e-4068-be64-5fc9794bc1bc"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.76510620117188,
    "Text": "as",
    "TextType": "PRINTED",
    "Geometry": {
      "BoundingBox": {
        "Width": 0.02041218988597393,
        "Height": 0.025104399770498276,
        "Left": 0.2207803726196289,
        "Top": 0.44124215841293335
      },
      "Polygon": [
        {
          "X": 0.2207803726196289,
          "Y": 0.44124215841293335
        },
        {
          "X": 0.24119256436824799,
          "Y": 0.44124215841293335
        },
        {
          "X": 0.24119256436824799,
          "Y": 0.4663465619087219
        },
        {
          "X": 0.2207803726196289,
          "Y": 0.4663465619087219
        }
      ]
    },
    "Id": "e96eb92c-6774-4d6f-8f4a-68a7618d4c66"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.9301528930664,
```

```
      "Text": "above",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.05268359184265137,
          "Height": 0.03216424956917763,
          "Left": 0.24375422298908234,
          "Top": 0.4354657828807831
        },
        "Polygon": [
          {
            "X": 0.24375422298908234,
            "Y": 0.4354657828807831
          },
          {
            "X": 0.2964377999305725,
            "Y": 0.4354657828807831
          },
          {
            "X": 0.2964377999305725,
            "Y": 0.4676300287246704
          },
          {
            "X": 0.24375422298908234,
            "Y": 0.4676300287246704
          }
        ]
      },
      "Id": "88b85c05-427a-4d4f-8cc4-3667234e8364"
    },
    {
      "BlockType": "WORD",
      "Confidence": 85.3905029296875,
      "Text": "Previous",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.09860499948263168,
          "Height": 0.04000622034072876,
          "Left": 0.3194798231124878,
          "Top": 0.5194430351257324
        },
        "Polygon": [
          {
```

```
            "X": 0.3194798231124878,
            "Y": 0.5194430351257324
          },
          {
            "X": 0.4180848002433777,
            "Y": 0.5194430351257324
          },
          {
            "X": 0.4180848002433777,
            "Y": 0.5594492554664612
          },
          {
            "X": 0.3194798231124878,
            "Y": 0.5594492554664612
          }
        ]
      },
      "Id": "8b324501-bf38-4ce9-9777-6514b7ade760"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.14524841308594,
      "Text": "Employment",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.14039960503578186,
          "Height": 0.04645847901701927,
          "Left": 0.4214291274547577,
          "Top": 0.5219109654426575
        },
        "Polygon": [
          {
            "X": 0.4214291274547577,
            "Y": 0.5219109654426575
          },
          {
            "X": 0.5618287324905396,
            "Y": 0.5219109654426575
          },
          {
            "X": 0.5618287324905396,
            "Y": 0.568369448184967
          },
```

```
        {
          "X": 0.4214291274547577,
          "Y": 0.568369448184967
        }
      ]
    },
    "Id": "b0cea99a-5045-464d-ac8a-a63ab0470995"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.48454284667969,
    "Text": "History",
    "TextType": "PRINTED",
    "Geometry": {
      "BoundingBox": {
        "Width": 0.08361124992370605,
        "Height": 0.05192042887210846,
        "Left": 0.5668527483940125,
        "Top": 0.5172380208969116
      },
      "Polygon": [
        {
          "X": 0.5668527483940125,
          "Y": 0.5172380208969116
        },
        {
          "X": 0.6504639983177185,
          "Y": 0.5172380208969116
        },
        {
          "X": 0.6504639983177185,
          "Y": 0.5691584348678589
        },
        {
          "X": 0.5668527483940125,
          "Y": 0.5691584348678589
        }
      ]
    },
    "Id": "b92a6ee5-ca59-44dc-9c47-534c133b11e7"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.78699493408203,
```

```
      "Text": "Start",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.041341401636600494,
          "Height": 0.030926469713449478,
          "Left": 0.034429505467414856,
          "Top": 0.6124123334884644
        },
        "Polygon": [
          {
            "X": 0.034429505467414856,
            "Y": 0.6124123334884644
          },
          {
            "X": 0.07577090710401535,
            "Y": 0.6124123334884644
          },
          {
            "X": 0.07577090710401535,
            "Y": 0.6433387994766235
          },
          {
            "X": 0.034429505467414856,
            "Y": 0.6433387994766235
          }
        ]
      },
      "Id": "ffe8b8e0-df59-4ac5-9aba-6b54b7c51b45"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.55198669433594,
      "Text": "Date",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.03923053666949272,
          "Height": 0.03072454035282135,
          "Left": 0.07830137014389038,
          "Top": 0.6123942136764526
        },
        "Polygon": [
          {
```

```
            "X": 0.07830137014389038,
            "Y": 0.6123942136764526
          },
          {
            "X": 0.1175319105386734,
            "Y": 0.6123942136764526
          },
          {
            "X": 0.1175319105386734,
            "Y": 0.6431187391281128
          },
          {
            "X": 0.07830137014389038,
            "Y": 0.6431187391281128
          }
        ]
      },
      "Id": "91e582cd-9871-4e9c-93cc-848baa426338"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.8897705078125,
      "Text": "End",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.03212086856365204,
          "Height": 0.03193363919854164,
          "Left": 0.14846202731132507,
          "Top": 0.6120467782020569
        },
        "Polygon": [
          {
            "X": 0.14846202731132507,
            "Y": 0.6120467782020569
          },
          {
            "X": 0.1805828958749771,
            "Y": 0.6120467782020569
          },
          {
            "X": 0.1805828958749771,
            "Y": 0.6439804434776306
          },
```

```
          {
            "X": 0.14846202731132507,
            "Y": 0.6439804434776306
          }
        ]
      },
      "Id": "7c97b56b-699f-49b0-93f4-98e6d90b107c"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.8445816040039,
      "Text": "Date",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.03987143933773041,
          "Height": 0.03142518177628517,
          "Left": 0.1844055950641632,
          "Top": 0.612853467464447
        },
        "Polygon": [
          {
            "X": 0.1844055950641632,
            "Y": 0.612853467464447
          },
          {
            "X": 0.22427703440189362,
            "Y": 0.612853467464447
          },
          {
            "X": 0.22427703440189362,
            "Y": 0.6442786455154419
          },
          {
            "X": 0.1844055950641632,
            "Y": 0.6442786455154419
          }
        ]
      },
      "Id": "7af04e27-0c15-447e-a569-b30edb99a133"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.9652328491211,
```

```
      "Text": "Employer",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08150768280029297,
          "Height": 0.0392492301762104,
          "Left": 0.2647075653076172,
          "Top": 0.6140711903572083
        },
        "Polygon": [
          {
            "X": 0.2647075653076172,
            "Y": 0.6140711903572083
          },
          {
            "X": 0.34621524810791016,
            "Y": 0.6140711903572083
          },
          {
            "X": 0.34621524810791016,
            "Y": 0.6533204317092896
          },
          {
            "X": 0.2647075653076172,
            "Y": 0.6533204317092896
          }
        ]
      },
      "Id": "a9bfeb55-75cd-47cd-b953-728e602a3564"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.94273376464844,
      "Text": "Name",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.05018233880400658,
          "Height": 0.03248906135559082,
          "Left": 0.34925445914268494,
          "Top": 0.6162016987800598
        },
        "Polygon": [
          {
```

```
            "X": 0.34925445914268494,
            "Y": 0.6162016987800598
          },
          {
            "X": 0.3994368016719818,
            "Y": 0.6162016987800598
          },
          {
            "X": 0.3994368016719818,
            "Y": 0.6486907601356506
          },
          {
            "X": 0.34925445914268494,
            "Y": 0.6486907601356506
          }
        ]
      },
      "Id": "9f0f9c06-d02c-4b07-bb39-7ade70be2c1b"
    },
    {
      "BlockType": "WORD",
      "Confidence": 98.85071563720703,
      "Text": "Position",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.07007700204849243,
          "Height": 0.03255689889192581,
          "Left": 0.49973347783088684,
          "Top": 0.6164342164993286
        },
        "Polygon": [
          {
            "X": 0.49973347783088684,
            "Y": 0.6164342164993286
          },
          {
            "X": 0.5698104500770569,
            "Y": 0.6164342164993286
          },
          {
            "X": 0.5698104500770569,
            "Y": 0.6489911079406738
          },
```

```
          {
            "X": 0.49973347783088684,
            "Y": 0.6489911079406738
          }
        ]
      },
      "Id": "6d5edf02-845c-40e0-9514-e56d0d652ae0"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.86096954345703,
      "Text": "Held",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.04017873853445053,
          "Height": 0.03292537108063698,
          "Left": 0.5734874606132507,
          "Top": 0.614840030670166
        },
        "Polygon": [
          {
            "X": 0.5734874606132507,
            "Y": 0.614840030670166
          },
          {
            "X": 0.6136662364006042,
            "Y": 0.614840030670166
          },
          {
            "X": 0.6136662364006042,
            "Y": 0.6477653980255127
          },
          {
            "X": 0.5734874606132507,
            "Y": 0.6477653980255127
          }
        ]
      },
      "Id": "3297ab59-b237-45fb-ae60-a108f0c95ac2"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.97740936279297,
```

```
      "Text": "Reason",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.06497219949960709,
          "Height": 0.03248770162463188,
          "Left": 0.7430596351623535,
          "Top": 0.6136704087257385
        },
        "Polygon": [
          {
            "X": 0.7430596351623535,
            "Y": 0.6136704087257385
          },
          {
            "X": 0.8080317974090576,
            "Y": 0.6136704087257385
          },
          {
            "X": 0.8080317974090576,
            "Y": 0.6461580991744995
          },
          {
            "X": 0.7430596351623535,
            "Y": 0.6461580991744995
          }
        ]
      },
      "Id": "f4b8cf26-d2da-4a76-8345-69562de3cc11"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.98371887207031,
      "Text": "for",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.029645200818777084,
          "Height": 0.03462234139442444,
          "Left": 0.8108851909637451,
          "Top": 0.6117717623710632
        },
        "Polygon": [
          {
```

```
            "X": 0.8108851909637451,
            "Y": 0.6117717623710632
          },
          {
            "X": 0.8405303955078125,
            "Y": 0.6117717623710632
          },
          {
            "X": 0.8405303955078125,
            "Y": 0.6463940739631653
          },
          {
            "X": 0.8108851909637451,
            "Y": 0.6463940739631653
          }
        ]
      },
      "Id": "386d4a63-1194-4c0e-a18d-4d074a0b1f93"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.98424530029297,
      "Text": "leaving",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.06517849862575531,
          "Height": 0.040626998990774155,
          "Left": 0.8430007100105286,
          "Top": 0.6116235852241516
        },
        "Polygon": [
          {
            "X": 0.8430007100105286,
            "Y": 0.6116235852241516
          },
          {
            "X": 0.9081792235374451,
            "Y": 0.6116235852241516
          },
          {
            "X": 0.9081792235374451,
            "Y": 0.6522505879402161
          },
```

```
          {
            "X": 0.8430007100105286,
            "Y": 0.6522505879402161
          }
        ]
      },
      "Id": "a8622541-1896-4d54-8d10-7da2c800ec5c"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.77413177490234,
      "Text": "1/15/2009",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08799663186073303,
          "Height": 0.03832906112074852,
          "Left": 0.03175082430243492,
          "Top": 0.691371738910675
        },
        "Polygon": [
          {
            "X": 0.03175082430243492,
            "Y": 0.691371738910675
          },
          {
            "X": 0.11974745243787766,
            "Y": 0.691371738910675
          },
          {
            "X": 0.11974745243787766,
            "Y": 0.7297008037567139
          },
          {
            "X": 0.03175082430243492,
            "Y": 0.7297008037567139
          }
        ]
      },
      "Id": "da7a6482-0964-49a4-bc7d-56942ff3b4e1"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.72286224365234,
```

```
      "Text": "6/30/2011",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08843102306127548,
          "Height": 0.03991425037384033,
          "Left": 0.14642837643623352,
          "Top": 0.6919752955436707
        },
        "Polygon": [
          {
            "X": 0.14642837643623352,
            "Y": 0.6919752955436707
          },
          {
            "X": 0.2348593920469284,
            "Y": 0.6919752955436707
          },
          {
            "X": 0.2348593920469284,
            "Y": 0.731889545917511
          },
          {
            "X": 0.14642837643623352,
            "Y": 0.731889545917511
          }
        ]
      },
      "Id": "5a8da66a-ecce-4ee9-a765-a46d6cdc6cde"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.92295837402344,
      "Text": "Any",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.034067559987306595,
          "Height": 0.037968240678310394,
          "Left": 0.2626699209213257,
          "Top": 0.6972727179527283
        },
        "Polygon": [
          {
```

```
            "X": 0.2626699209213257,
            "Y": 0.6972727179527283
          },
          {
            "X": 0.2967374622821808,
            "Y": 0.6972727179527283
          },
          {
            "X": 0.2967374622821808,
            "Y": 0.7352409362792969
          },
          {
            "X": 0.2626699209213257,
            "Y": 0.7352409362792969
          }
        ]
      },
      "Id": "77749c2b-aa7f-450e-8dd2-62bcaf253ba2"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.81578063964844,
      "Text": "Company",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08160992711782455,
          "Height": 0.03890080004930496,
          "Left": 0.29906952381134033,
          "Top": 0.6978086829185486
        },
        "Polygon": [
          {
            "X": 0.29906952381134033,
            "Y": 0.6978086829185486
          },
          {
            "X": 0.3806794583797455,
            "Y": 0.6978086829185486
          },
          {
            "X": 0.3806794583797455,
            "Y": 0.736709475517273
          },
```

```
        {
          "X": 0.29906952381134033,
          "Y": 0.736709475517273
        }
      ]
    },
    "Id": "713bad19-158d-4e3e-b01f-f5707ddb04e5"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.37964630126953,
    "Text": "Assistant",
    "TextType": "PRINTED",
    "Geometry": {
      "BoundingBox": {
        "Width": 0.0789310410618782,
        "Height": 0.03139699995517731,
        "Left": 0.49814170598983765,
        "Top": 0.7005078196525574
      },
      "Polygon": [
        {
          "X": 0.49814170598983765,
          "Y": 0.7005078196525574
        },
        {
          "X": 0.5770727396011353,
          "Y": 0.7005078196525574
        },
        {
          "X": 0.5770727396011353,
          "Y": 0.7319048047065735
        },
        {
          "X": 0.49814170598983765,
          "Y": 0.7319048047065735
        }
      ]
    },
    "Id": "989944f9-f684-4714-87d8-9ad9a321d65c"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.784912109375,
```

```
      "Text": "baker",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.050264399498701096,
          "Height": 0.03237773850560188,
          "Left": 0.5806865096092224,
          "Top": 0.699238657951355
        },
        "Polygon": [
          {
            "X": 0.5806865096092224,
            "Y": 0.699238657951355
          },
          {
            "X": 0.630950927734375,
            "Y": 0.699238657951355
          },
          {
            "X": 0.630950927734375,
            "Y": 0.7316163778305054
          },
          {
            "X": 0.5806865096092224,
            "Y": 0.7316163778305054
          }
        ]
      },
      "Id": "ae82e2aa-1601-4e0c-8340-1db7ad0c9a31"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.96180725097656,
      "Text": "relocated",
      "TextType": "PRINTED",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08668994158506393,
          "Height": 0.03330250084400177,
          "Left": 0.7426905632019043,
          "Top": 0.6974037289619446
        },
        "Polygon": [
          {
```

```
            "X": 0.7426905632019043,
            "Y": 0.6974037289619446
          },
          {
            "X": 0.8293805122375488,
            "Y": 0.6974037289619446
          },
          {
            "X": 0.8293805122375488,
            "Y": 0.7307062149047852
          },
          {
            "X": 0.7426905632019043,
            "Y": 0.7307062149047852
          }
        ]
      },
      "Id": "a9cf9a8c-fdaa-413e-9346-5a28a98aebdb"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.98190307617188,
      "Text": "7/1/2011",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.09747002273797989,
          "Height": 0.07067439705133438,
          "Left": 0.028500309213995934,
          "Top": 0.7745237946510315
        },
        "Polygon": [
          {
            "X": 0.028500309213995934,
            "Y": 0.7745237946510315
          },
          {
            "X": 0.12597033381462097,
            "Y": 0.7745237946510315
          },
          {
            "X": 0.12597033381462097,
            "Y": 0.8451982140541077
          },
```

```
        {
          "X": 0.028500309213995934,
          "Y": 0.8451982140541077
        }
      ]
    },
    "Id": "0f711065-1872-442a-ba6d-8fababaa452a"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.98418426513672,
    "Text": "8/10/2013",
    "TextType": "HANDWRITING",
    "Geometry": {
      "BoundingBox": {
        "Width": 0.10664612054824829,
        "Height": 0.06439515948295593,
        "Left": 0.14159755408763885,
        "Top": 0.7791688442230225
      },
      "Polygon": [
        {
          "X": 0.14159755408763885,
          "Y": 0.7791688442230225
        },
        {
          "X": 0.24824367463588715,
          "Y": 0.7791688442230225
        },
        {
          "X": 0.24824367463588715,
          "Y": 0.843563973903656
        },
        {
          "X": 0.14159755408763885,
          "Y": 0.843563973903656
        }
      ]
    },
    "Id": "a92d8eef-db28-45ba-801a-5da0f589d277"
  },
  {
    "BlockType": "WORD",
    "Confidence": 99.97722625732422,
```

```
      "Text": "Example",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.12127546221017838,
          "Height": 0.05682983994483948,
          "Left": 0.26764172315597534,
          "Top": 0.794414758682251
        },
        "Polygon": [
          {
            "X": 0.26764172315597534,
            "Y": 0.794414758682251
          },
          {
            "X": 0.3889172077178955,
            "Y": 0.794414758682251
          },
          {
            "X": 0.3889172077178955,
            "Y": 0.8512446284294128
          },
          {
            "X": 0.26764172315597534,
            "Y": 0.8512446284294128
          }
        ]
      },
      "Id": "d6962efb-34ab-4ffb-9f2f-5f263e813558"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.98429870605469,
      "Text": "Corp.",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.07650306820869446,
          "Height": 0.05481306090950966,
          "Left": 0.4026312530040741,
          "Top": 0.7980174422264099
        },
        "Polygon": [
          {
```

```
            "X": 0.4026312530040741,
            "Y": 0.7980174422264099
          },
          {
            "X": 0.47913432121276855,
            "Y": 0.7980174422264099
          },
          {
            "X": 0.47913432121276855,
            "Y": 0.8528305292129517
          },
          {
            "X": 0.4026312530040741,
            "Y": 0.8528305292129517
          }
        ]
      },
      "Id": "1876c8ea-d3e8-4c39-870e-47512b3b5080"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.91166687011719,
      "Text": "Baker",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.09931197017431259,
          "Height": 0.06008723005652428,
          "Left": 0.5098910331726074,
          "Top": 0.787897527217865
        },
        "Polygon": [
          {
            "X": 0.5098910331726074,
            "Y": 0.787897527217865
          },
          {
            "X": 0.609203040599823,
            "Y": 0.787897527217865
          },
          {
            "X": 0.609203040599823,
            "Y": 0.8479847311973572
          },
```

```
          {
            "X": 0.5098910331726074,
            "Y": 0.8479847311973572
          }
        ]
      },
      "Id": "00adeaef-ed57-44eb-b8a9-503575236d62"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.98870849609375,
      "Text": "better",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.10782185196876526,
          "Height": 0.06207133084535599,
          "Left": 0.7428008317947388,
          "Top": 0.7928366661071777
        },
        "Polygon": [
          {
            "X": 0.7428008317947388,
            "Y": 0.7928366661071777
          },
          {
            "X": 0.8506226539611816,
            "Y": 0.7928366661071777
          },
          {
            "X": 0.8506226539611816,
            "Y": 0.8549079895019531
          },
          {
            "X": 0.7428008317947388,
            "Y": 0.8549079895019531
          }
        ]
      },
      "Id": "c0fc9a58-7a4b-4f69-bafd-2cff32be2665"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.8883285522461,
```

```
      "Text": "opp.",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.07421936094760895,
          "Height": 0.058906231075525284,
          "Left": 0.8577775359153748,
          "Top": 0.8038780689239502
        },
        "Polygon": [
          {
            "X": 0.8577775359153748,
            "Y": 0.8038780689239502
          },
          {
            "X": 0.9319969415664673,
            "Y": 0.8038780689239502
          },
          {
            "X": 0.9319969415664673,
            "Y": 0.8627843260765076
          },
          {
            "X": 0.8577775359153748,
            "Y": 0.8627843260765076
          }
        ]
      },
      "Id": "bf6dc8ee-2fb3-4b6c-aee4-31e96912a2d8"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.92573547363281,
      "Text": "8/15/2013",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.10257463902235031,
          "Height": 0.05412459000945091,
          "Left": 0.027909137308597565,
          "Top": 0.8608770370483398
        },
        "Polygon": [
          {
```

```
            "X": 0.027909137308597565,
            "Y": 0.8608770370483398
          },
          {
            "X": 0.13048377633094788,
            "Y": 0.8608770370483398
          },
          {
            "X": 0.13048377633094788,
            "Y": 0.915001630783081
          },
          {
            "X": 0.027909137308597565,
            "Y": 0.915001630783081
          }
        ]
      },
      "Id": "5384f860-f857-4a94-9438-9dfa20eed1c6"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.99625396728516,
      "Text": "Present",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.09982697665691376,
          "Height": 0.06888339668512344,
          "Left": 0.1420602649450302,
          "Top": 0.8511748909950256
        },
        "Polygon": [
          {
            "X": 0.1420602649450302,
            "Y": 0.8511748909950256
          },
          {
            "X": 0.24188724160194397,
            "Y": 0.8511748909950256
          },
          {
            "X": 0.24188724160194397,
            "Y": 0.9200583100318909
          },
```

```
          {
            "X": 0.1420602649450302,
            "Y": 0.9200583100318909
          }
        ]
      },
      "Id": "0bb96ed6-b2e6-4da4-90b3-b85561bbd89d"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.9826431274414,
      "Text": "AnyCompany",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.18611273169517517,
          "Height": 0.08581399917602539,
          "Left": 0.2615866959095001,
          "Top": 0.869536280632019
        },
        "Polygon": [
          {
            "X": 0.2615866959095001,
            "Y": 0.869536280632019
          },
          {
            "X": 0.4476994276046753,
            "Y": 0.869536280632019
          },
          {
            "X": 0.4476994276046753,
            "Y": 0.9553502798080444
          },
          {
            "X": 0.2615866959095001,
            "Y": 0.9553502798080444
          }
        ]
      },
      "Id": "25343360-d906-440a-88b7-92eb89e95949"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.99523162841797,
```

```
      "Text": "head",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.07429949939250946,
          "Height": 0.05485520139336586,
          "Left": 0.49359121918678284,
          "Top": 0.8714361190795898
        },
        "Polygon": [
          {
            "X": 0.49359121918678284,
            "Y": 0.8714361190795898
          },
          {
            "X": 0.5678907036781311,
            "Y": 0.8714361190795898
          },
          {
            "X": 0.5678907036781311,
            "Y": 0.926291286945343
          },
          {
            "X": 0.49359121918678284,
            "Y": 0.926291286945343
          }
        ]
      },
      "Id": "0ef3c194-8322-4575-94f1-82819ee57e3a"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.99574279785156,
      "Text": "baker",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.1019822508096695,
          "Height": 0.05615599825978279,
          "Left": 0.585354208946228,
          "Top": 0.8702592849731445
        },
        "Polygon": [
          {
```

```
            "X": 0.585354208946228,
            "Y": 0.8702592849731445
          },
          {
            "X": 0.6873364448547363,
            "Y": 0.8702592849731445
          },
          {
            "X": 0.6873364448547363,
            "Y": 0.9264153242111206
          },
          {
            "X": 0.585354208946228,
            "Y": 0.9264153242111206
          }
        ]
      },
      "Id": "d296acd9-3e9a-4985-95f8-f863614f2c46"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.9880599975586,
      "Text": "N/A,",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.08230073750019073,
          "Height": 0.06588289886713028,
          "Left": 0.7411766648292542,
          "Top": 0.8722732067108154
        },
        "Polygon": [
          {
            "X": 0.7411766648292542,
            "Y": 0.8722732067108154
          },
          {
            "X": 0.8234773874282837,
            "Y": 0.8722732067108154
          },
          {
            "X": 0.8234773874282837,
            "Y": 0.9381561279296875
          },
```

```
          {
            "X": 0.7411766648292542,
            "Y": 0.9381561279296875
          }
        ]
      },
      "Id": "195cfb5b-ae06-4203-8520-4e4b0a73b5ce"
    },
    {
      "BlockType": "WORD",
      "Confidence": 99.97914123535156,
      "Text": "current",
      "TextType": "HANDWRITING",
      "Geometry": {
        "BoundingBox": {
          "Width": 0.12791454792022705,
          "Height": 0.04768490046262741,
          "Left": 0.8387037515640259,
          "Top": 0.8843405842781067
        },
        "Polygon": [
          {
            "X": 0.8387037515640259,
            "Y": 0.8843405842781067
          },
          {
            "X": 0.9666182994842529,
            "Y": 0.8843405842781067
          },
          {
            "X": 0.9666182994842529,
            "Y": 0.9320254921913147
          },
          {
            "X": 0.8387037515640259,
            "Y": 0.9320254921913147
          }
        ]
      },
      "Id": "549ef3f9-3a13-4b77-bc25-fb2e0996839a"
    }
  ],
  "DetectDocumentTextModelVersion": "1.0",
  "ResponseMetadata": {
```

```
    "RequestId": "337129e6-3af7-4014-842b-f6484e82cbf6",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "x-amzn-requestid": "337129e6-3af7-4014-842b-f6484e82cbf6",
      "content-type": "application/x-amz-json-1.1",
      "content-length": "45675",
      "date": "Mon, 09 Nov 2020 23:54:38 GMT"
    },
    "RetryAttempts": 0
  }
}
}
```

使用 Amazon Textract 檢測文檔文本

若要檢測文檔中的文本，請使用DetectDocumentText操作，並將文檔文件作為輸入傳遞。DetectDocumentText返回一個 JSON 結構，其中包含檢測到的文本的行和單詞、文本在文檔中的位置以及檢測到的文本之間的關係。如需詳細資訊，請參閱 偵測文字。

您提供的輸入文檔可以是影像位元組陣列 (base64 編碼影像位元組)，或者 Amazon S3 物件。在此步驟中，您會將影像檔案上傳至您的 S3 儲存貯體，並指定檔案名稱。

若要檢測文檔中的文字 (API)

1. 如果您尚未：

   a. 使用創建或更新 IAM 用户AmazonTextractFullAccess和AmazonS3ReadOnlyAccess許可。如需詳細資訊，請參閱 步驟 1：設定 AWS 帳户並建立 IAM 使用者。

   b. 安裝並設定 AWS CLI 和 AWS SDK。如需詳細資訊，請參閱 步驟 2：設定AWS CLI和AWS開發套件。

2. 將檔案上傳至 S3 儲存貯體。

   如需說明，請參閱「」將物件上傳至 Amazon S3中的Amazon Simple Storage Service 用户指南。

3. 使用下列範例來呼叫 DetectDocumentText 操作。

   Java

   以下示例代碼在檢測到的文本行周圍顯示文檔和框。

在函數中main中，替換bucket和document以您在步驟 2 中所使用的 Amazon S3 儲存貯體名稱與文件名稱來取代和。

```
//Calls DetectDocumentText.
//Loads document from S3 bucket. Displays the document and bounding boxes around
 detected lines/words of text.
package com.amazonaws.samples;

import java.awt.*;
import java.awt.image.BufferedImage;
import java.util.List;
import javax.imageio.ImageIO;
import javax.swing.*;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.S3ObjectInputStream;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.amazonaws.services.textract.AmazonTextract;
import com.amazonaws.services.textract.AmazonTextractClientBuilder;
import com.amazonaws.services.textract.model.Block;
import com.amazonaws.services.textract.model.BoundingBox;
import com.amazonaws.services.textract.model.DetectDocumentTextRequest;
import com.amazonaws.services.textract.model.DetectDocumentTextResult;
import com.amazonaws.services.textract.model.Document;
import com.amazonaws.services.textract.model.S3Object;
import com.amazonaws.services.textract.model.Point;
import com.amazonaws.services.textract.model.Relationship;

public class DocumentText extends JPanel {

    private static final long serialVersionUID = 1L;

    BufferedImage image;
    DetectDocumentTextResult result;

    public DocumentText(DetectDocumentTextResult documentResult, BufferedImage
 bufImage) throws Exception {
        super();

        result = documentResult; // Results of text detection.
        image = bufImage; // The image containing the document.
```

```
   }

   // Draws the image and text bounding box.
   public void paintComponent(Graphics g) {

       int height = image.getHeight(this);
       int width = image.getWidth(this);

       Graphics2D g2d = (Graphics2D) g; // Create a Java2D version of g.

       // Draw the image.
       g2d.drawImage(image, 0, 0, image.getWidth(this) , image.getHeight(this),
this);

       // Iterate through blocks and display polygons around lines of detected
text.
       List<Block> blocks = result.getBlocks();
       for (Block block : blocks) {
           DisplayBlockInfo(block);
           if ((block.getBlockType()).equals("LINE")) {
               ShowPolygon(height, width, block.getGeometry().getPolygon(),
g2d);
               /*
                 ShowBoundingBox(height, width,
block.getGeometry().getBoundingBox(), g2d);
                */
           } else { // its a word, so just show vertical lines.
               ShowPolygonVerticals(height, width,
block.getGeometry().getPolygon(), g2d);
           }
       }
   }

   // Show bounding box at supplied location.
   private void ShowBoundingBox(int imageHeight, int imageWidth, BoundingBox
box, Graphics2D g2d) {

       float left = imageWidth * box.getLeft();
       float top = imageHeight * box.getTop();

       // Display bounding box.
       g2d.setColor(new Color(0, 212, 0));
       g2d.drawRect(Math.round(left), Math.round(top),
```

```
                Math.round(imageWidth * box.getWidth()), Math.round(imageHeight
* box.getHeight()));

   }

   // Shows polygon at supplied location
   private void ShowPolygon(int imageHeight, int imageWidth, List<Point>
points, Graphics2D g2d) {

       g2d.setColor(new Color(0, 0, 0));
       Polygon polygon = new Polygon();

       // Construct polygon and display
       for (Point point : points) {
           polygon.addPoint((Math.round(point.getX() * imageWidth)),
                   Math.round(point.getY() * imageHeight));
       }
       g2d.drawPolygon(polygon);
   }

   // Draws only the vertical lines in the supplied polygon.
   private void ShowPolygonVerticals(int imageHeight, int imageWidth,
List<Point> points, Graphics2D g2d) {

       g2d.setColor(new Color(0, 212, 0));
       Object[] parry = points.toArray();
       g2d.setStroke(new BasicStroke(2));

       g2d.drawLine(Math.round(((Point) parry[0]).getX() * imageWidth),
               Math.round(((Point) parry[0]).getY() * imageHeight),
Math.round(((Point) parry[3]).getX() * imageWidth),
               Math.round(((Point) parry[3]).getY() * imageHeight));

       g2d.setColor(new Color(255, 0, 0));
       g2d.drawLine(Math.round(((Point) parry[1]).getX() * imageWidth),
               Math.round(((Point) parry[1]).getY() * imageHeight),
Math.round(((Point) parry[2]).getX() * imageWidth),
               Math.round(((Point) parry[2]).getY() * imageHeight));

   }
   //Displays information from a block returned by text detection and text
analysis
   private void DisplayBlockInfo(Block block) {
       System.out.println("Block Id : " + block.getId());
```

```
        if (block.getText()!=null)
            System.out.println("    Detected text: " + block.getText());
        System.out.println("    Type: " + block.getBlockType());

        if (block.getBlockType().equals("PAGE") !=true) {
            System.out.println("    Confidence: " +
block.getConfidence().toString());
        }
        if(block.getBlockType().equals("CELL"))
        {
            System.out.println("    Cell information:");
            System.out.println("        Column: " + block.getColumnIndex());
            System.out.println("        Row: " + block.getRowIndex());
            System.out.println("        Column span: " + block.getColumnSpan());
            System.out.println("        Row span: " + block.getRowSpan());

        }

        System.out.println("    Relationships");
        List<Relationship> relationships=block.getRelationships();
        if(relationships!=null) {
            for (Relationship relationship : relationships) {
                System.out.println("        Type: " + relationship.getType());
                System.out.println("        IDs: " +
relationship.getIds().toString());
            }
        } else {
            System.out.println("        No related Blocks");
        }

        System.out.println("    Geometry");
        System.out.println("        Bounding Box: " +
block.getGeometry().getBoundingBox().toString());
        System.out.println("        Polygon: " +
block.getGeometry().getPolygon().toString());

        List<String> entityTypes = block.getEntityTypes();

        System.out.println("    Entity Types");
        if(entityTypes!=null) {
            for (String entityType : entityTypes) {
                System.out.println("        Entity Type: " + entityType);
            }
        } else {
```

```
            System.out.println("        No entity type");
        }
        if(block.getPage()!=null)
            System.out.println("    Page: " + block.getPage());
        System.out.println();
    }

    public static void main(String arg[]) throws Exception {

        // The S3 bucket and document
        String document = "";
        String bucket = "";


        AmazonS3 s3client = AmazonS3ClientBuilder.standard()
                .withEndpointConfiguration(
                        new EndpointConfiguration("https://
s3.amazonaws.com","us-east-1"))
                .build();


        // Get the document from S3
        com.amazonaws.services.s3.model.S3Object s3object =
 s3client.getObject(bucket, document);
        S3ObjectInputStream inputStream = s3object.getObjectContent();
        BufferedImage image = ImageIO.read(inputStream);

        // Call DetectDocumentText
        EndpointConfiguration endpoint = new EndpointConfiguration(
                "https://textract.us-east-1.amazonaws.com", "us-east-1");
        AmazonTextract client = AmazonTextractClientBuilder.standard()
                .withEndpointConfiguration(endpoint).build();


        DetectDocumentTextRequest request = new DetectDocumentTextRequest()
            .withDocument(new Document().withS3Object(new
 S3Object().withName(document).withBucket(bucket)));

        DetectDocumentTextResult result = client.detectDocumentText(request);

        // Create frame and panel.
        JFrame frame = new JFrame("RotateImage");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        DocumentText panel = new DocumentText(result, image);
```

```
        panel.setPreferredSize(new Dimension(image.getWidth() ,
 image.getHeight() ));
        frame.setContentPane(panel);
        frame.pack();
        frame.setVisible(true);

    }
}
```

AWS CLI

此 AWS CLI 命令顯示 detect-document-text CLI 操作的 JSON 輸出。

替換Bucket和Name以您在步驟 2 中所使用的 Amazon S3 儲存貯體名稱與文件名稱來取代和。

```
aws textract detect-document-text \
 --document '{"S3Object":{"Bucket":"bucket","Name":"document"}}'
```

Python

以下範例程式碼會顯示在偵測到的文本行周圍的文件和框。

在函數中main中，替換bucket和document以您在步驟 2 中所使用的 Amazon S3 儲存貯體名稱與文件名稱來取代和。

```
#Detects text in a document stored in an S3 bucket. Display polygon box around
 text and angled text
import boto3
import io
from io import BytesIO
import sys

import psutil
import time

import math
from PIL import Image, ImageDraw, ImageFont


# Displays information about a block returned by text detection and text
 analysis
```

```
def DisplayBlockInformation(block):
    print('Id: {}'.format(block['Id']))
    if 'Text' in block:
        print('    Detected: ' + block['Text'])
    print('    Type: ' + block['BlockType'])

    if 'Confidence' in block:
        print('    Confidence: ' + "{:.2f}".format(block['Confidence']) + "%")

    if block['BlockType'] == 'CELL':
        print("    Cell information")
        print("        Column: " + str(block['ColumnIndex']))
        print("        Row: " + str(block['RowIndex']))
        print("        ColumnSpan: " + str(block['ColumnSpan']))
        print("        RowSpan: " + str(block['RowSpan']))

    if 'Relationships' in block:
        print('    Relationships: {}'.format(block['Relationships']))
    print('    Geometry: ')
    print('        Bounding Box: {}'.format(block['Geometry']['BoundingBox']))
    print('        Polygon: {}'.format(block['Geometry']['Polygon']))

    if block['BlockType'] == "KEY_VALUE_SET":
        print ('    Entity Type: ' + block['EntityTypes'][0])
    if 'Page' in block:
        print('Page: ' + block['Page'])
    print()

def process_text_detection(bucket, document):


    #Get the document from S3
    s3_connection = boto3.resource('s3')

    s3_object = s3_connection.Object(bucket,document)
    s3_response = s3_object.get()

    stream = io.BytesIO(s3_response['Body'].read())
    image=Image.open(stream)


    # Detect text in the document

    client = boto3.client('textract')
```

```
    #process using image bytes
    #image_binary = stream.getvalue()
    #response = client.detect_document_text(Document={'Bytes': image_binary})

    #process using S3 object
    response = client.detect_document_text(
        Document={'S3Object': {'Bucket': bucket, 'Name': document}})

    #Get the text blocks
    blocks=response['Blocks']
    width, height =image.size
    draw = ImageDraw.Draw(image)
    print ('Detected Document Text')

    # Create image showing bounding box/polygon the detected lines/text
    for block in blocks:
            print('Type: ' + block['BlockType'])
            if block['BlockType'] != 'PAGE':
                print('Detected: ' + block['Text'])
                print('Confidence: ' + "{:.2f}".format(block['Confidence']) +
"%")

            print('Id: {}'.format(block['Id']))
            if 'Relationships' in block:
                print('Relationships: {}'.format(block['Relationships']))
            print('Bounding Box: {}'.format(block['Geometry']['BoundingBox']))
            print('Polygon: {}'.format(block['Geometry']['Polygon']))
            print()
            draw=ImageDraw.Draw(image)
            # Draw WORD - Green -  start of word, red - end of word
            if block['BlockType'] == "WORD":
                draw.line([(width * block['Geometry']['Polygon'][0]['X'],
                height * block['Geometry']['Polygon'][0]['Y']),
                (width * block['Geometry']['Polygon'][3]['X'],
                height * block['Geometry']['Polygon'][3]['Y'])],fill='green',
                width=2)

                draw.line([(width * block['Geometry']['Polygon'][1]['X'],
                height * block['Geometry']['Polygon'][1]['Y']),
                (width * block['Geometry']['Polygon'][2]['X'],
                height * block['Geometry']['Polygon'][2]['Y'])],
                fill='red',
                width=2)
```

```
            # Draw box around entire LINE
            if block['BlockType'] == "LINE":
                points=[]

                for polygon in block['Geometry']['Polygon']:
                    points.append((width * polygon['X'], height * polygon['Y']))

                draw.polygon((points), outline='black')

                # Uncomment to draw bounding box
                #box=block['Geometry']['BoundingBox']
                #left = width * box['Left']
                #top = height * box['Top']
                #draw.rectangle([left,top, left + (width * box['Width']), top
 +(height * box['Height'])],outline='black')


    # Display the image
    image.show()
    # display image for 10 seconds


    return len(blocks)

def main():

    bucket = ''
    document = ''
    block_count=process_text_detection(bucket,document)
    print("Blocks detected: " + str(block_count))

if __name__ == "__main__":
    main()
```

Node.js

以下 Node.js 示例代碼顯示檢測到的文本行周圍的文檔和框，從而將結果的圖像輸出到運行代碼的目錄。它利用`image-size`和`images`套件。

在函數中`main`中，替換`bucket`和`document`以您在步驟 2 中所使用的 Amazon S3 儲存貯體名稱與文件名稱來取代和。替換`regionConfig`以及您賬户所在區域的名稱。

```
async function main(){

// Import AWS
const AWS = require("aws-sdk")
// Use Image-Size to get
const sizeOf = require('image-size');
// Image tool to draw buffers
const images = require("images");

// Create a canvas and get the context
const { createCanvas } = require('canvas')
const canvas = createCanvas(200, 200)
const ctx = canvas.getContext('2d')

// Set variables
const bucket = 'bucket-name' // the s3 bucket name
const photo  = 'image-name' // the name of file
const regionConfig = 'region'

// Set region if needed
AWS.config.update({region:regionConfig});

// Connect to Textract
const client = new AWS.Textract();
// Connect to S3 to display image
const s3 = new AWS.S3();

// Define paramaters
const params = {
  Document: {
    S3Object: {
      Bucket: bucket,
      Name: photo
    },
  },
}

// Function to display image
async function getImage(){
  const imageData =  s3.getObject(
    {
        Bucket: bucket,
        Key: photo
```

```
      }

  ).promise();
  return imageData;
}

// get image
var imageData = await getImage()

// Get the height, width of the image
const dimensions = sizeOf(imageData.Body)
const width = dimensions.width
const height = dimensions.height
console.log(imageData.Body)
console.log(width, height)

canvas.width = width;
canvas.height = height;

try{
  // Call API and log response
  const res = await client.detectDocumentText(params).promise();
  var image = images(imageData.Body).size(width, height)
  //console.log the type of block, text, text type, and confidence
  res.Blocks.forEach(block => {
    console.log(`Block Type: ${block.BlockType}`),
    console.log(`Text: ${block.Text}`)
    console.log(`TextType: ${block.TextType}`)
    console.log(`Confidence: ${block.Confidence}`)

    // Draw box around detected text using polygons
    ctx.strokeStyle = 'rgba(0,0,0,0.5)';
    ctx.beginPath();
    block.Geometry.Polygon.forEach(({X, Y}) =>
    ctx.lineTo(width * X - 10, height * Y - 10)
    );
    ctx.closePath();
    ctx.stroke();
    console.log("-----")
  })

  // render image
  var buffer = canvas.toBuffer("image/png");
  image.draw(images(buffer), 10, 10)
```

```
  image.save("output-image.jpg");

} catch (err){
console.error(err);}

}

main()
```

4. 執行範例。Python 和 Java 示例顯示了文檔圖像。每一行檢測到的文本周圍都有一個黑框。綠色的垂直線是檢測到的單詞的開頭。紅色垂直線是檢測到的單詞的結尾。所以此AWS CLI示例僅顯示`DetectDocumentText`operation.

使用 Amazon Textract 分析文檔文本

若要分析文檔中的文本，請使用AnalyzeDocument操作，並將文檔文件作為輸入傳遞。`AnalyzeDocument`會傳回含有分析文字的 JSON 結構。如需詳細資訊，請參閱 分析文檔。

您提供的輸入文檔可以是影像位元組陣列 (base64 編碼影像位元組)，或者 Amazon S3 物件。在此步驟中，您會將影像檔案上傳至您的 S3 儲存貯體，並指定檔案名稱。

分析文檔中的文本 (API)

1. 如果您尚未：

   a. 使用創建或更新 IAM 用户`AmazonTextractFullAccess`和`AmazonS3ReadOnlyAccess`許可。如需詳細資訊，請參閱 步驟 1：設定 AWS 帳户並建立 IAM 使用者。

   b. 安裝並設定 AWS CLI 和 AWS SDK。如需詳細資訊，請參閱 步驟 2：設定AWS CLI和AWS開發套件。

2. 將含有文檔的影像上傳至您的 S3 儲存貯體。

   如需說明，請參閱「」將物件上傳至 Amazon S3中的Amazon Simple Storage Service 用户指南。

3. 使用下列範例來呼叫 `AnalyzeDocument` 操作。

   Java

   以下範例程式碼會顯示檢測到的項目周圍的文件和框。

在函數中main中，替換bucket和document以您在步驟 2 中所使用的 Amazon S3 儲存貯體名稱與文件影像名稱來取代和。

```
//Loads document from S3 bucket. Displays the document and polygon around
 detected lines of text.
package com.amazonaws.samples;

import java.awt.*;
import java.awt.image.BufferedImage;
import java.util.List;
import javax.imageio.ImageIO;
import javax.swing.*;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.S3ObjectInputStream;
import com.amazonaws.services.textract.AmazonTextract;
import com.amazonaws.services.textract.AmazonTextractClientBuilder;
import com.amazonaws.services.textract.model.AnalyzeDocumentRequest;
import com.amazonaws.services.textract.model.AnalyzeDocumentResult;
import com.amazonaws.services.textract.model.Block;
import com.amazonaws.services.textract.model.BoundingBox;
import com.amazonaws.services.textract.model.Document;
import com.amazonaws.services.textract.model.S3Object;
import com.amazonaws.services.textract.model.Point;
import com.amazonaws.services.textract.model.Relationship;
import com.amazonaws.client.builder.AwsClientBuilder.EndpointConfiguration;

public class AnalyzeDocument extends JPanel {

    private static final long serialVersionUID = 1L;

    BufferedImage image;

    AnalyzeDocumentResult result;

    public AnalyzeDocument(AnalyzeDocumentResult documentResult, BufferedImage
 bufImage) throws Exception {
        super();

        result = documentResult; // Results of text detection.
        image = bufImage; // The image containing the document.
```

```
   }

   // Draws the image and text bounding box.
   public void paintComponent(Graphics g) {

       int height = image.getHeight(this);
       int width = image.getWidth(this);

       Graphics2D g2d = (Graphics2D) g; // Create a Java2D version of g.

       // Draw the image.
       g2d.drawImage(image, 0, 0, image.getWidth(this), image.getHeight(this),
this);

       // Iterate through blocks and display bounding boxes around everything.

       List<Block> blocks = result.getBlocks();
       for (Block block : blocks) {
           DisplayBlockInfo(block);
           switch(block.getBlockType()) {

           case "KEY_VALUE_SET":
               if (block.getEntityTypes().contains("KEY")){
                   ShowBoundingBox(height, width,
block.getGeometry().getBoundingBox(), g2d, new Color(255,0,0));
               }
               else {  //VALUE
                   ShowBoundingBox(height, width,
block.getGeometry().getBoundingBox(), g2d, new Color(0,255,0));
               }
               break;
           case "TABLE":
               ShowBoundingBox(height, width,
block.getGeometry().getBoundingBox(), g2d, new Color(0,0,255));
               break;
           case "CELL":
               ShowBoundingBox(height, width,
block.getGeometry().getBoundingBox(), g2d, new Color(255,255,0));
               break;
           case "SELECTION_ELEMENT":
               if (block.getSelectionStatus().equals("SELECTED"))
                   ShowSelectedElement(height, width,
block.getGeometry().getBoundingBox(), g2d, new Color(0,0,255));
```

```
                break;
            default:
                //PAGE, LINE & WORD
                //ShowBoundingBox(height, width,
block.getGeometry().getBoundingBox(), g2d, new Color(200,200,0));
            }
        }

         // uncomment to show polygon around all blocks
         //ShowPolygon(height,width,block.getGeometry().getPolygon(),g2d);


    }

    // Show bounding box at supplied location.
    private void ShowBoundingBox(int imageHeight, int imageWidth, BoundingBox
box, Graphics2D g2d, Color color) {

        float left = imageWidth * box.getLeft();
        float top = imageHeight * box.getTop();

        // Display bounding box.
        g2d.setColor(color);
        g2d.drawRect(Math.round(left), Math.round(top),
                Math.round(imageWidth * box.getWidth()), Math.round(imageHeight
* box.getHeight()));

    }
    private void ShowSelectedElement(int imageHeight, int imageWidth,
BoundingBox box, Graphics2D g2d, Color color) {

        float left = imageWidth * box.getLeft();
        float top = imageHeight * box.getTop();

        // Display bounding box.
        g2d.setColor(color);
        g2d.fillRect(Math.round(left), Math.round(top),
                Math.round(imageWidth * box.getWidth()), Math.round(imageHeight
* box.getHeight()));

    }

    // Shows polygon at supplied location
```

```
   private void ShowPolygon(int imageHeight, int imageWidth, List<Point>
points, Graphics2D g2d) {

       g2d.setColor(new Color(0, 0, 0));
       Polygon polygon = new Polygon();

       // Construct polygon and display
       for (Point point : points) {
           polygon.addPoint((Math.round(point.getX() * imageWidth)),
                   Math.round(point.getY() * imageHeight));
       }
       g2d.drawPolygon(polygon);
   }
   //Displays information from a block returned by text detection and text
analysis
   private void DisplayBlockInfo(Block block) {
       System.out.println("Block Id : " + block.getId());
       if (block.getText()!=null)
           System.out.println("    Detected text: " + block.getText());
       System.out.println("    Type: " + block.getBlockType());

       if (block.getBlockType().equals("PAGE") !=true) {
           System.out.println("    Confidence: " +
block.getConfidence().toString());
       }
       if(block.getBlockType().equals("CELL"))
       {
           System.out.println("    Cell information:");
           System.out.println("        Column: " + block.getColumnIndex());
           System.out.println("        Row: " + block.getRowIndex());
           System.out.println("        Column span: " + block.getColumnSpan());
           System.out.println("        Row span: " + block.getRowSpan());

       }

       System.out.println("    Relationships");
       List<Relationship> relationships=block.getRelationships();
       if(relationships!=null) {
           for (Relationship relationship : relationships) {
               System.out.println("        Type: " + relationship.getType());
               System.out.println("        IDs: " +
relationship.getIds().toString());
           }
       } else {
```

```
            System.out.println("        No related Blocks");
        }

        System.out.println("    Geometry");
        System.out.println("        Bounding Box: " +
 block.getGeometry().getBoundingBox().toString());
        System.out.println("        Polygon: " +
 block.getGeometry().getPolygon().toString());

        List<String> entityTypes = block.getEntityTypes();

        System.out.println("    Entity Types");
        if(entityTypes!=null) {
            for (String entityType : entityTypes) {
                System.out.println("        Entity Type: " + entityType);
            }
        } else {
            System.out.println("        No entity type");
        }

        if(block.getBlockType().equals("SELECTION_ELEMENT")) {
            System.out.print("    Selection element detected: ");
            if (block.getSelectionStatus().equals("SELECTED")){
                System.out.println("Selected");
            }else {
                System.out.println(" Not selected");
            }
        }

        if(block.getPage()!=null)
            System.out.println("    Page: " + block.getPage());
        System.out.println();
    }

    public static void main(String arg[]) throws Exception {

        // The S3 bucket and document
        String document = "";
        String bucket = "";

        AmazonS3 s3client = AmazonS3ClientBuilder.standard()
                .withEndpointConfiguration(
                        new EndpointConfiguration("https://
s3.amazonaws.com","us-east-1"))
```

```
                .build();


        // Get the document from S3
        com.amazonaws.services.s3.model.S3Object s3object =
 s3client.getObject(bucket, document);
        S3ObjectInputStream inputStream = s3object.getObjectContent();
        BufferedImage image = ImageIO.read(inputStream);

        // Call AnalyzeDocument
        EndpointConfiguration endpoint = new EndpointConfiguration(
                "https://textract.us-east-1.amazonaws.com", "us-east-1");
        AmazonTextract client = AmazonTextractClientBuilder.standard()
                .withEndpointConfiguration(endpoint).build();


        AnalyzeDocumentRequest request = new AnalyzeDocumentRequest()
                .withFeatureTypes("TABLES","FORMS")
                 .withDocument(new Document().
                       withS3Object(new
 S3Object().withName(document).withBucket(bucket)));


        AnalyzeDocumentResult result = client.analyzeDocument(request);

        // Create frame and panel.
        JFrame frame = new JFrame("RotateImage");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        AnalyzeDocument panel = new AnalyzeDocument(result, image);
        panel.setPreferredSize(new Dimension(image.getWidth(),
 image.getHeight()));
        frame.setContentPane(panel);
        frame.pack();
        frame.setVisible(true);

    }
}
```

AWS CLI

此 AWS CLI 命令顯示 `detect-document-text` CLI 操作的 JSON 輸出。

替換Bucket和Name以您在步驟 2 中所使用的 Amazon S3 儲存貯體名稱與文件名稱來取代和。

```
aws textract analyze-document \
 --document '{"S3Object":{"Bucket":"bucket","Name":"document"}}' \
 --feature-types '["TABLES","FORMS"]'
```

Python

以下範例程式碼會顯示檢測到的項目周圍的文件和框。

在函數中main中，替換bucket和document以您在步驟 2 中所使用的 Amazon S3 儲存貯體名稱與文件名稱來取代和。

```
#Analyzes text in a document stored in an S3 bucket. Display polygon box around
 text and angled text
import boto3
import io
from io import BytesIO
import sys

import math
from PIL import Image, ImageDraw, ImageFont

def ShowBoundingBox(draw,box,width,height,boxColor):

    left = width * box['Left']
    top = height * box['Top']
    draw.rectangle([left,top, left + (width * box['Width']), top +(height *
 box['Height'])],outline=boxColor)

def ShowSelectedElement(draw,box,width,height,boxColor):

    left = width * box['Left']
    top = height * box['Top']
    draw.rectangle([left,top, left + (width * box['Width']), top +(height *
 box['Height'])],fill=boxColor)

# Displays information about a block returned by text detection and text
 analysis
def DisplayBlockInformation(block):
```

```
    print('Id: {}'.format(block['Id']))
    if 'Text' in block:
        print('    Detected: ' + block['Text'])
    print('    Type: ' + block['BlockType'])

    if 'Confidence' in block:
        print('    Confidence: ' + "{:.2f}".format(block['Confidence']) + "%")

    if block['BlockType'] == 'CELL':
        print("    Cell information")
        print("        Column:" + str(block['ColumnIndex']))
        print("        Row:" + str(block['RowIndex']))
        print("        Column Span:" + str(block['ColumnSpan']))
        print("        RowSpan:" + str(block['ColumnSpan']))

    if 'Relationships' in block:
        print('    Relationships: {}'.format(block['Relationships']))
    print('    Geometry: ')
    print('        Bounding Box: {}'.format(block['Geometry']['BoundingBox']))
    print('        Polygon: {}'.format(block['Geometry']['Polygon']))

    if block['BlockType'] == "KEY_VALUE_SET":
        print ('    Entity Type: ' + block['EntityTypes'][0])

    if block['BlockType'] == 'SELECTION_ELEMENT':
        print('    Selection element detected: ', end='')

        if block['SelectionStatus'] =='SELECTED':
            print('Selected')
        else:
            print('Not selected')

    if 'Page' in block:
        print('Page: ' + block['Page'])
    print()

def process_text_analysis(bucket, document):

    #Get the document from S3
    s3_connection = boto3.resource('s3')

    s3_object = s3_connection.Object(bucket,document)
    s3_response = s3_object.get()
```

```
    stream = io.BytesIO(s3_response['Body'].read())
    image=Image.open(stream)

    # Analyze the document
    client = boto3.client('textract')

    image_binary = stream.getvalue()
    response = client.analyze_document(Document={'Bytes': image_binary},
        FeatureTypes=["TABLES", "FORMS"])

    ### Alternatively, process using S3 object ###
    #response = client.analyze_document(
    #    Document={'S3Object': {'Bucket': bucket, 'Name': document}},
    #    FeatureTypes=["TABLES", "FORMS"])

    ### To use a local file ###
    # with open("pathToFile", 'rb') as img_file:
        ### To display image using PIL ###
    #    image = Image.open()
        ### Read bytes ###
    #    img_bytes = img_file.read()
    #    response = client.analyze_document(Document={'Bytes': img_bytes},
 FeatureTypes=["TABLES", "FORMS"])


    #Get the text blocks
    blocks=response['Blocks']
    width, height =image.size
    draw = ImageDraw.Draw(image)
    print ('Detected Document Text')

    # Create image showing bounding box/polygon the detected lines/text
    for block in blocks:

        DisplayBlockInformation(block)

        draw=ImageDraw.Draw(image)
        if block['BlockType'] == "KEY_VALUE_SET":
            if block['EntityTypes'][0] == "KEY":
                ShowBoundingBox(draw, block['Geometry']
['BoundingBox'],width,height,'red')
            else:
                ShowBoundingBox(draw, block['Geometry']
['BoundingBox'],width,height,'green')
```

```
        if block['BlockType'] == 'TABLE':
            ShowBoundingBox(draw, block['Geometry']['BoundingBox'],width,height,
 'blue')

        if block['BlockType'] == 'CELL':
            ShowBoundingBox(draw, block['Geometry']['BoundingBox'],width,height,
 'yellow')
        if block['BlockType'] == 'SELECTION_ELEMENT':
            if block['SelectionStatus'] =='SELECTED':
                ShowSelectedElement(draw, block['Geometry']
['BoundingBox'],width,height, 'blue')

            #uncomment to draw polygon for all Blocks
            #points=[]
            #for polygon in block['Geometry']['Polygon']:
            #    points.append((width * polygon['X'], height * polygon['Y']))
            #draw.polygon((points), outline='blue')

    # Display the image
    image.show()
    return len(blocks)


def main():

    bucket = ''
    document = ''
    block_count=process_text_analysis(bucket,document)
    print("Blocks detected: " + str(block_count))

if __name__ == "__main__":
    main()
```

Node.js

以下範例程式碼會顯示檢測到的項目周圍的文件和框。

在下面的代碼中，替換bucket和photo以您在步驟 2 中所使用的 Amazon S3 儲存貯體名稱與文件名稱來取代和。替換region以及與帳户相關聯的地區。

```
// Import required AWS SDK clients and commands for Node.js
import { AnalyzeDocumentCommand } from  "@aws-sdk/client-textract";
```

```
import  { TextractClient } from "@aws-sdk/client-textract";

// Set the AWS Region.
const REGION = "region"; //e.g. "us-east-1"
// Create SNS service object.
const textractClient = new TextractClient({ region: REGION });

const bucket = 'buckets'
const photo = 'photo'

// Set params
const params = {
    Document: {
      S3Object: {
        Bucket: bucket,
        Name: photo
      },
    },
    FeatureTypes: ['TABLES', 'FORMS'],
  }

const displayBlockInfo = async (response) => {
    try {
        response.Blocks.forEach(block => {
            console.log(`ID: ${block.Id}`)
            console.log(`Block Type: ${block.BlockType}`)
            if ("Text" in block && block.Text !== undefined){
                console.log(`Text: ${block.Text}`)
            }
            else{}
            if ("Confidence" in block && block.Confidence !== undefined){
                console.log(`Confidence: ${block.Confidence}`)
            }
            else{}
            if (block.BlockType == 'CELL'){
                console.log("Cell info:")
                console.log(`   Column Index - ${block.ColumnIndex}`)
                console.log(`   Row - ${block.RowIndex}`)
                console.log(`   Column Span - ${block.ColumnSpan}`)
                console.log(`   Row Span - ${block.RowSpan}`)
            }
            if ("Relationships" in block && block.Relationships !== undefined){
                console.log(block.Relationships)
                console.log("Geometry:")
```

```
                console.log(`   Bounding Box -
 ${JSON.stringify(block.Geometry.BoundingBox)}`)
                console.log(`   Polygon -
 ${JSON.stringify(block.Geometry.Polygon)}`)
            }
            console.log("-----")
        });
      } catch (err) {
        console.log("Error", err);
      }
}

const analyze_document_text = async () => {
    try {
        const analyzeDoc = new AnalyzeDocumentCommand(params);
        const response = await textractClient.send(analyzeDoc);
        //console.log(response)
        displayBlockInfo(response)
        return response; // For unit tests.
      } catch (err) {
        console.log("Error", err);
      }
}

analyze_document_text()
```

4. 執行範例。Python 和 Java 示例顯示帶有以下彩色邊界框的文檔圖像：

   - 紅色 — 密鑰塊對象
   - 綠色 — 值塊對象
   - 藍色 — 表格塊對象
   - 黃色 — 單元格塊對象

   選取的選擇元素將用藍色填充。

   所以此AWS CLI示例僅顯示`AnalyzeDocument`operation.

使用 Amazon Textract 分析發票和收據

要分析發票和收據單據，您可以使用分析費用 API，並傳遞文檔文件作為輸入。AnalyzeExpense是返回包含分析文本的 JSON 結構的同步操作。如需詳細資訊，請參閱 分析發票和收款。

要異步分析發票和收據，請使用StartExpenseAnalysis開始處理輸入文檔文件並使用GetExpenseAnalysis取得結果。

您提供的輸入文檔可以是影像位元組陣列 (base64 編碼影像位元組)，或者 Amazon S3 物件。在此步驟中，您會將影像檔案上傳至您的 S3 儲存貯體，並指定檔案名稱。

分析發票或收據 (API)

1. 如果您尚未：
   a. 使用創建或更新 IAM 用户AmazonTextractFullAccess和AmazonS3ReadOnlyAccess許可。如需詳細資訊，請參閱 步驟 1：設定 AWS 帳户並建立 IAM 使用者。
   b. 安裝並設定 AWS CLI 和 AWS SDK。如需詳細資訊，請參閱 步驟 2：設定AWS CLI和AWS開發套件。
2. 將含有文檔的影像上傳至您的 S3 儲存貯體。

   如需說明，請參閱「」將物件上傳至 Amazon S3中的Amazon Simple Storage Service 用户指南。
3. 使用下列範例來呼叫 AnalyzeExpense 操作。

   CLI

```
  aws textract analyze-expense --document '{"S3Object": {"Bucket": "bucket
 name","Name":
"object name"}}'
```

   Python

```
import boto3
import io
from PIL import Image, ImageDraw

def draw_bounding_box(key, val, width, height, draw):
```

```
    # If a key is Geometry, draw the bounding box info in it
    if "Geometry" in key:
        # Draw bounding box information
        box = val["BoundingBox"]
        left = width * box['Left']
        top = height * box['Top']
        draw.rectangle([left, top, left + (width * box['Width']), top + (height
 * box['Height'])],
                       outline='black')

# Takes a field as an argument and prints out the detected labels and values
def print_labels_and_values(field):
    # Only if labels are detected and returned
    if "LabelDetection" in field:
        print("Summary Label Detection - Confidence: {}".format(
            str(field.get("LabelDetection")["Confidence"])) + ", "
              + "Summary Values: {}".format(str(field.get("LabelDetection")
["Text"])))
        print(field.get("LabelDetection")["Geometry"])
    else:
        print("Label Detection - No labels returned.")
    if "ValueDetection" in field:
        print("Summary Value Detection - Confidence: {}".format(
            str(field.get("ValueDetection")["Confidence"])) + ", "
              + "Summary Values: {}".format(str(field.get("ValueDetection")
["Text"])))
        print(field.get("ValueDetection")["Geometry"])
    else:
        print("Value Detection - No values returned")

def process_text_detection(bucket, document):
    # Get the document from S3
    s3_connection = boto3.resource('s3')
    s3_object = s3_connection.Object(bucket, document)
    s3_response = s3_object.get()

    # opening binary stream using an in-memory bytes buffer
    stream = io.BytesIO(s3_response['Body'].read())

    # loading stream into image
    image = Image.open(stream)

    # Detect text in the document
    client = boto3.client('textract', region_name="us-east-1")
```

```
    # process using S3 object
    response = client.analyze_expense(
        Document={'S3Object': {'Bucket': bucket, 'Name': document}})

    # Set width and height to display image and draw bounding boxes
    # Create drawing object
    width, height = image.size
    draw = ImageDraw.Draw(image)

    for expense_doc in response["ExpenseDocuments"]:
        for line_item_group in expense_doc["LineItemGroups"]:
            for line_items in line_item_group["LineItems"]:
                for expense_fields in line_items["LineItemExpenseFields"]:
                    print_labels_and_values(expense_fields)
                    print()

        print("Summary:")
        for summary_field in expense_doc["SummaryFields"]:
            print_labels_and_values(summary_field)
            print()

        #For draw bounding boxes
        for line_item_group in expense_doc["LineItemGroups"]:
            for line_items in line_item_group["LineItems"]:
                for expense_fields in line_items["LineItemExpenseFields"]:
                    for key, val in expense_fields["ValueDetection"].items():
                        if "Geometry" in key:
                            draw_bounding_box(key, val, width, height, draw)

        for label in expense_doc["SummaryFields"]:
            if "LabelDetection" in label:
                for key, val in label["LabelDetection"].items():
                    draw_bounding_box(key, val, width, height, draw)

    # Display the image
    image.show()

def main():
    bucket = 'Bucket-Name'
    document = 'Document-Name'
    process_text_detection(bucket, document)

if __name__ == "__main__":
```

```
    main()
```

Java

```
package com.amazonaws.samples;

import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import javax.imageio.ImageIO;
import javax.swing.*;
import software.amazon.awssdk.auth.credentials.AwsBasicCredentials;
import software.amazon.awssdk.auth.credentials.StaticCredentialsProvider;
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.async.AsyncResponseTransformer;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.*;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.AnalyzeExpenseRequest;
import software.amazon.awssdk.services.textract.model.AnalyzeExpenseResponse;
import software.amazon.awssdk.services.textract.model.BoundingBox;
import software.amazon.awssdk.services.textract.model.Document;
import software.amazon.awssdk.services.textract.model.ExpenseDocument;
import software.amazon.awssdk.services.textract.model.ExpenseField;
import software.amazon.awssdk.services.textract.model.LineItemFields;
import software.amazon.awssdk.services.textract.model.LineItemGroup;
import software.amazon.awssdk.services.textract.model.S3Object;
import software.amazon.awssdk.services.textract.model.Point;

/**
 *
 * Demo code to parse Textract AnalyzeExpense API
 *
 */
public class TextractAnalyzeExpenseSample extends JPanel {
```

```
private static final long serialVersionUID = 1L;

BufferedImage image;
static AnalyzeExpenseResponse result;

public TextractAnalyzeExpenseSample(AnalyzeExpenseResponse documentResult,
BufferedImage bufImage) throws Exception {
 super();

 result = documentResult; // Results of analyzeexpense summaryfields and
lineitemgroups detection.
 image = bufImage; // The image containing the document.

}

// Draws the image and text bounding box.
public void paintComponent(Graphics g) {

 Graphics2D g2d = (Graphics2D) g; // Create a Java2D version of g.

 // Draw the image.
 g2d.drawImage(image, 0, 0, image.getWidth(this), image.getHeight(this), this);

 // Iterate through summaryfields and lineitemgroups and display boundedboxes
around lines of detected label and value.
 List<ExpenseDocument> expenseDocuments = result.expenseDocuments();
 for (ExpenseDocument expenseDocument : expenseDocuments) {

  if (expenseDocument.hasSummaryFields()) {
   DisplayAnalyzeExpenseSummaryInfo(expenseDocument);
   List<ExpenseField> summaryfields = expenseDocument.summaryFields();
   for (ExpenseField summaryfield : summaryfields) {

    if (summaryfield.valueDetection() != null) {
     ShowBoundingBox(image.getHeight(this), image.getWidth(this),
       summaryfield.valueDetection().geometry().boundingBox(), g2d, new
Color(0, 0, 0));
    }

    if (summaryfield.labelDetection() != null) {

     ShowBoundingBox(image.getHeight(this), image.getWidth(this),
```

```
        summaryfield.labelDetection().geometry().boundingBox(), g2d, new
Color(0, 0, 0));

    }

   }

  }

  if (expenseDocument.hasLineItemGroups()) {
   DisplayAnalyzeExpenseLineItemGroupsInfo(expenseDocument);

   List<LineItemGroup> lineitemgroups = expenseDocument.lineItemGroups();

   for (LineItemGroup lineitemgroup : lineitemgroups) {

    if (lineitemgroup.hasLineItems()) {

     List<LineItemFields> lineItems = lineitemgroup.lineItems();
     for (LineItemFields lineitemfield : lineItems) {

      if (lineitemfield.hasLineItemExpenseFields()) {

       List<ExpenseField> expensefields =
lineitemfield.lineItemExpenseFields();
       for (ExpenseField expensefield : expensefields) {

        if (expensefield.valueDetection() != null) {
         ShowBoundingBox(image.getHeight(this), image.getWidth(this),
           expensefield.valueDetection().geometry().boundingBox(), g2d,
           new Color(0, 0, 0));
        }

        if (expensefield.labelDetection() != null) {
         ShowBoundingBox(image.getHeight(this), image.getWidth(this),
           expensefield.labelDetection().geometry().boundingBox(), g2d,
           new Color(0, 0, 0));
        }

       }
      }

     }
```

```
    }

   }

  }
 }

}

// Show bounding box at supplied location.
private void ShowBoundingBox(float imageHeight, float imageWidth, BoundingBox
box, Graphics2D g2d, Color color) {

 float left = imageWidth * box.left();
 float top = imageHeight * box.top();

 // Display bounding box.
 g2d.setColor(color);
 g2d.drawRect(Math.round(left), Math.round(top), Math.round(imageWidth *
box.width()),
   Math.round(imageHeight * box.height()));

}

private void ShowSelectedElement(float imageHeight, float imageWidth,
BoundingBox box, Graphics2D g2d,
  Color color) {

 float left = (float) imageWidth * (float) box.left();
 float top = (float) imageHeight * (float) box.top();
 System.out.println(left);
 System.out.println(top);

 // Display bounding box.
 g2d.setColor(color);
 g2d.fillRect(Math.round(left), Math.round(top), Math.round(imageWidth *
box.width()),
   Math.round(imageHeight * box.height()));

}

// Shows polygon at supplied location
private void ShowPolygon(int imageHeight, int imageWidth, List<Point> points,
Graphics2D g2d) {
```

```
 g2d.setColor(new Color(0, 0, 0));
 Polygon polygon = new Polygon();

 // Construct polygon and display
 for (Point point : points) {
  polygon.addPoint((Math.round(point.x() * imageWidth)), Math.round(point.y() *
imageHeight));
 }
 g2d.drawPolygon(polygon);
}

private void DisplayAnalyzeExpenseSummaryInfo(ExpenseDocument expensedocument)
{
 System.out.println(" ExpenseId : " + expensedocument.expenseIndex());
 System.out.println("    Expense Summary information:");
 if (expensedocument.hasSummaryFields()) {

  List<ExpenseField> summaryfields = expensedocument.summaryFields();

  for (ExpenseField summaryfield : summaryfields) {

   System.out.println("    Page: " + summaryfield.pageNumber());
   if (summaryfield.type() != null) {

    System.out.println("    Expense Summary Field Type:" +
summaryfield.type().text());

   }
   if (summaryfield.labelDetection() != null) {

    System.out.println("    Expense Summary Field Label:" +
summaryfield.labelDetection().text());
    System.out.println("    Geometry");
    System.out.println("        Bounding Box: "
      + summaryfield.labelDetection().geometry().boundingBox().toString());
    System.out.println(
      "        Polygon: " +
summaryfield.labelDetection().geometry().polygon().toString());

   }
   if (summaryfield.valueDetection() != null) {
    System.out.println("    Expense Summary Field Value:" +
summaryfield.valueDetection().text());
```

```
    System.out.println("    Geometry");
    System.out.println("        Bounding Box: "
      + summaryfield.valueDetection().geometry().boundingBox().toString());
    System.out.println(
      "        Polygon: " +
summaryfield.valueDetection().geometry().polygon().toString());

   }

  }

 }

}

private void DisplayAnalyzeExpenseLineItemGroupsInfo(ExpenseDocument
expensedocument) {

 System.out.println(" ExpenseId : " + expensedocument.expenseIndex());
 System.out.println("    Expense LineItemGroups information:");

 if (expensedocument.hasLineItemGroups()) {

  List<LineItemGroup> lineitemgroups = expensedocument.lineItemGroups();

  for (LineItemGroup lineitemgroup : lineitemgroups) {

   System.out.println("    Expense LineItemGroupsIndexID :" +
lineitemgroup.lineItemGroupIndex());

   if (lineitemgroup.hasLineItems()) {

    List<LineItemFields> lineItems = lineitemgroup.lineItems();

    for (LineItemFields lineitemfield : lineItems) {

     if (lineitemfield.hasLineItemExpenseFields()) {

      List<ExpenseField> expensefields = lineitemfield.lineItemExpenseFields();
      for (ExpenseField expensefield : expensefields) {

       if (expensefield.type() != null) {
        System.out.println("    Expense LineItem Field Type:" +
expensefield.type().text());
```

```
        }

        if (expensefield.valueDetection() != null) {
         System.out.println(
           "    Expense Summary Field Value:" +
expensefield.valueDetection().text());
         System.out.println("    Geometry");
         System.out.println("        Bounding Box: "
           + expensefield.valueDetection().geometry().boundingBox().toString());
         System.out.println("        Polygon: "
           + expensefield.valueDetection().geometry().polygon().toString());

        }

        if (expensefield.labelDetection() != null) {
         System.out.println(
           "    Expense LineItem Field Label:" +
expensefield.labelDetection().text());
         System.out.println("    Geometry");
         System.out.println("        Bounding Box: "
           + expensefield.labelDetection().geometry().boundingBox().toString());
         System.out.println("        Polygon: "
           + expensefield.labelDetection().geometry().polygon().toString());
        }

       }
      }

     }

    }
  }
 }

public static void main(String arg[]) throws Exception {

 // Creates a default async client with credentials and AWS Region loaded from
 // the
 // environment
```

```
  S3AsyncClient client =
 S3AsyncClient.builder().region(Region.US_EAST_1).build();

  System.out.println("Creating the S3 Client");

  // Start the call to Amazon S3, not blocking to wait for the result
  CompletableFuture<ResponseBytes<GetObjectResponse>> responseFuture =
 client.getObject(
    GetObjectRequest.builder().bucket("textractanalyzeexpense").key("input/
sample-receipt.jpg").build(),
    AsyncResponseTransformer.toBytes());

  System.out.println("Successfully read the object");

  // When future is complete (either successfully or in error), handle the
  // response
  CompletableFuture<ResponseBytes<GetObjectResponse>> operationCompleteFuture =
 responseFuture
    .whenComplete((getObjectResponse, exception) -> {
     if (getObjectResponse != null) {
      // At this point, the file my-file.out has been created with the data
      // from S3; let's just print the object version
      // Convert this into Async call and remove the below block from here and
 put it
      // outside


      TextractClient textractclient =
 TextractClient.builder().region(Region.US_EAST_1).build();

      AnalyzeExpenseRequest request = AnalyzeExpenseRequest.builder()
        .document(
          Document.builder().s3Object(S3Object.builder().name("YOURObjectName")
            .bucket("YOURBucket").build()).build())
        .build();

      AnalyzeExpenseResponse result = textractclient.analyzeExpense(request);

      System.out.print(result.toString());

      ByteArrayInputStream bais = new
 ByteArrayInputStream(getObjectResponse.asByteArray());
      try {
       BufferedImage image = ImageIO.read(bais);
```

```
       System.out.println("Successfully read the image");
       JFrame frame = new JFrame("Expense Image");
       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
       TextractAnalyzeExpense panel = new TextractAnalyzeExpense(result, image);
       panel.setPreferredSize(new Dimension(image.getWidth(),
 image.getHeight()));
       frame.setContentPane(panel);
       frame.pack();
       frame.setVisible(true);
      } catch (IOException e) {
       throw new RuntimeException(e);
      } catch (Exception e) {
       // TODO Auto-generated catch block
       e.printStackTrace();
      }
     } else {
      // Handle the error
      exception.printStackTrace();
     }
    });

  // We could do other work while waiting for the AWS call to complete in
  // the background, but we'll just wait for "whenComplete" to finish instead
  operationCompleteFuture.join();

 }
}
```

Node.Js

```
                    // Import required AWS SDK clients and commands for Node.js
import { AnalyzeExpenseCommand } from  "@aws-sdk/client-textract";
import  { TextractClient } from "@aws-sdk/client-textract";

// Set the AWS Region.
const REGION = "region"; //e.g. "us-east-1"
// Create SNS service object.
const textractClient = new TextractClient({ region: REGION });

const bucket = 'bucket'
```

```
const photo = 'photo'

// Set params
const params = {
    Document: {
      S3Object: {
        Bucket: bucket,
        Name: photo
      },
    },
  }

const process_text_detection = async () => {
    try {
        const aExpense = new AnalyzeExpenseCommand(params);
        const response = await textractClient.send(aExpense);
        //console.log(response)
        response.ExpenseDocuments.forEach(doc => {
            doc.LineItemGroups.forEach(items => {
                items.LineItems.forEach(fields => {
                    fields.LineItemExpenseFields.forEach(expenseFields =>{
                        console.log(expenseFields)
                    })
                }
                )}
                )
            }
        )
        return response; // For unit tests.
      } catch (err) {
        console.log("Error", err);
      }
}

process_text_detection()
```

4. 這將為您提供`AnalyzeExpense`operation.

使用 Amazon Textract 分析身份文檔

要分析身份文檔，請使用 AnalyzeID API，並將文檔文件作為輸入傳遞。AnalyzeID會傳回含有分析文字的 JSON 結構。如需詳細資訊，請參閱 分析身分文件。

您提供的輸入文檔可以是影像位元組陣列 (base64 編碼影像位元組)，或者 Amazon S3 物件。在此步驟中，您會將影像檔案上傳至您的 S3 儲存貯體，並指定檔案名稱。

分析身份文檔 (API)

1. 如果您尚未：

   a. 使用創建或更新 IAM 用户AmazonTextractFullAccess和AmazonS3ReadOnlyAccess許可。如需詳細資訊，請參閱 步驟 1：設定 AWS 帳户並建立 IAM 使用者。

   b. 安裝並設定 AWS CLI 和 AWS SDK。如需詳細資訊，請參閱 步驟 2：設定AWS CLI和AWS開發套件。

2. 將含有文檔的影像上傳至您的 S3 儲存貯體。

   如需說明，請參閱「」將物件上傳至 Amazon S3中的Amazon Simple Storage Service 用户指南。

3. 使用下列範例來呼叫 AnalyzeID 操作。

   CLI

   下列範例從 S3 儲存貯體傳入文件，並運行AnalyzeID操作。在下面的代碼中，替換#使用 S3 儲存貯體的名稱，則##添加儲存貯體中的檔案名稱，以及##的名稱region與您的帳户相關聯。

```
aws textract analyze-id --document-pages '[{"S3Object":
{"Bucket":"bucket","Name":"name"}}]' --region region
```

   您也可以通過向輸入添加另一個 S3 對象，使用駕駛執照的正面和背面調用 API。

```
aws textract analyze-id --document-pages '[{"S3Object":
{"Bucket":"bucket","Name":"name front"}}, {"S3Object":
{"Bucket":"bucket","Name":"name back"}}]' --region us-east-1
```

如果您正在 Windows 設備上訪問 CLI，請使用雙引號而不是單引號，並用反斜槓（即\）轉義內部雙引號以解決您可能遇到的任何解析器錯誤。舉例來講，請參閱下列內容：

```
aws textract analyze-id --document-pages "[{\"S3Object\":{\"Bucket\":\"bucket\",
\"Name\":\"name\"}}]" --region region
```

Python

下列範例從 S3 儲存貯體傳入文件，並運行AnalyzeID操作，返回檢測到的鍵值組。在下面的代碼中，替換*bucket_name*與您 S3 儲存貯體的名稱一起使用，*###*添加儲存貯體中的檔案名稱，以及*##*的名稱region與您的帳戶相關聯。

```
import boto3

bucket_name = "bucket-name"
file_name = "file-name"
region = "region-name"

def analyze_id(region, bucket_name, file_name):

    textract_client = boto3.client('textract', region_name=region)
    response = textract_client.analyze_id(DocumentPages=[{"S3Object":
{"Bucket":bucket_name,"Name":file_name}}])

    for doc_fields in response['IdentityDocuments']:
        for id_field in doc_fields['IdentityDocumentFields']:
            for key, val in id_field.items():
                if "Type" in str(key):
                    print("Type: " + str(val['Text']))
            for key, val in id_field.items():
                if "ValueDetection" in str(key):
                    print("Value Detection: " + str(val['Text']))
            print()

analyze_id(region, bucket_name, file_name)
```

Java

下列範例從 S3 儲存貯體傳入文件，並運行AnalyzeID操作，返回檢測到的數據。在函數 main 中，將s3bucket和sourceDoc以您在步驟 2 中所使用的 Amazon S3 儲存貯體名稱與文件影像名稱來取代和。

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.amazonaws.samples;

import com.amazonaws.regions.Regions;
import com.amazonaws.services.textract.AmazonTextractClient;
import com.amazonaws.services.textract.AmazonTextractClientBuilder;
import com.amazonaws.services.textract.model.*;
import java.util.ArrayList;
import java.util.List;

public class AnalyzeIdentityDocument {

    public static void main(String[] args) {

        final String USAGE = "\n" +
                "Usage:\n" +
                "    <s3bucket><sourceDoc> \n\n" +
                "Where:\n" +
                "    s3bucket - the Amazon S3 bucket where the document is
 located. \n" +
                "    sourceDoc - the name of the document. \n";

        if (args.length != 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String s3bucket = "bucket-name"; //args[0];
        String sourceDoc = "sourcedoc-name";  //args[1];
        AmazonTextractClient textractClient = (AmazonTextractClient)
 AmazonTextractClientBuilder.standard()
                .withRegion(Regions.US_EAST_1)
```

```
                .build();

        getDocDetails(textractClient, s3bucket, sourceDoc);
    }

    public static void getDocDetails(AmazonTextractClient textractClient, String
 s3bucket, String sourceDoc ) {

       try {

            S3Object s3 = new S3Object();
            s3.setBucket(s3bucket);
            s3.setName(sourceDoc);

            com.amazonaws.services.textract.model.Document myDoc = new
 com.amazonaws.services.textract.model.Document();
            myDoc.setS3Object(s3);

            List<Document> list1 = new ArrayList();
            list1.add(myDoc);

            AnalyzeIDRequest idRequest = new AnalyzeIDRequest();
            idRequest.setDocumentPages(list1);

            AnalyzeIDResult result = textractClient.analyzeID(idRequest);
            List<IdentityDocument> docs =  result.getIdentityDocuments();
            for (IdentityDocument doc: docs) {

                List<IdentityDocumentField>idFields =
 doc.getIdentityDocumentFields();
                for (IdentityDocumentField field: idFields) {
                    System.out.println("Field type is "+
 field.getType().getText());
                    System.out.println("Field value is "+
 field.getValueDetection().getText());
                }
            }

       } catch (Exception e) {
            e.printStackTrace();
       }
    }
}
```

4. 這將為您提供`AnalyzeID`operation.

使用異步操作處理文檔

Amazon Textract 可以檢測和分析 PDF 或 TIFF 格式的多頁文檔中的文本。這包括發票和收據。多頁文件處理是一種非同步操作。異步處理文檔對於處理大型多頁文檔非常有用。例如，處理超過 1,000 頁的 PDF 文件需要一段時間。異步處理 PDF 文件允許您的應用程序在等待過程完成的同時完成其他任務。

本節介紹瞭如何使用 Amazon Textract 異步檢測和分析多頁或單頁文檔上的文本。多頁文檔必須採用 PDF 或 TIFF 格式。使用異步操作處理的單頁文檔可以採用 JPEG、PNG、TIFF 或 PDF 格式。

您可以將 Amazon Textract 異步操作用於以下目的：

- 文本檢測 — 您可以檢測多頁文檔上的行和單詞。異步操作是StartDocumentTextDetection和GetDocumentTextDetection。如需詳細資訊，請參閱 偵測文字。
- 文本分析 — 您可以識別多頁文檔中檢測到的文本之間的關係。異步操作是StartDocumentAnalysis和GetDocumentAnalysis。如需詳細資訊，請參閱 分析文檔。
- 費用分析 — 您可以識別多頁發票和收據上的數據關係。Amazon Textract 將每張發票或多頁文檔的收據頁面視為單獨收據或發票。它不會保留多頁文檔的一個頁面到另一個頁面的上下文。異步操作是StartExpenseAnalysis和GetExpenseAnalysis。如需詳細資訊，請參閱 分析發票和收款。

主題

調用 Amazon Textract 異步操作

Amazon Textract 提供了一個異步 API，您可以使用它來處理 PDF 或 TIFF 格式的多頁文檔。您還可以使用異步操作來處理 JPEG、PNG、TIFF 或 PDF 格式的單頁文檔。

本主題中的信息使用文本檢測操作來展示如何使用 Amazon Textract 異步操作。同樣的方法適用於the section called “StartDocumentAnalysis”和the section called “GetDocumentAnalysis”。它也適用於the section called “StartExpenseAnalysis”和the section called “GetExpenseAnalysis”。

如需範例，請參閱 檢測或分析多頁文檔中的文本。

Amazon Textract 以非同步方式處理存放在 Amazon S3 儲存貯體中的檔案。您可以通過調用`Start`操作，例如StartDocumentTextDetection。請求的完成狀態將發佈至 Amazon SNS Simple Notification Service (Amazon SNS) 主題。若要自 Amazon SNS 主題取得完成狀態，您可以使用 Amazon Simple Queue Service (Amazon SQS) 排列或AWS Lambda函數。在獲得完成狀態後，可呼叫如 GetDocumentTextDetection 的 `Get` 操作來取得請求結果。

默認情況下，異步調用的結果 Amazon S3 被加密並存儲在 Amazon Textract 擁有的存儲桶中 7 天，除非您使用操作的`OutputConfig`引數。

下表顯示了 Amazon Textract 支持的不同異步處理類型的相應「開始」和「獲取」操作：

啟動/取得亞馬遜文件異步操作的API 操作

處理類型	啟動 API	取得 API
文字偵測	StartDocumentTextDetection	GetDocumentTextDetection
文字分析	StartDocumentAnalysis	GetDocumentAnalysis
費用分析	起始筆分析	地理軟件分析

對於使用AWS Lambda函數，請參使用 Amazon Textract 進行大規模文檔處理。

下圖顯示在 Amazon S3 儲存貯體中的檔案映像中偵測檔案的程序。在圖表內，Amazon SQS 佇列將自 Amazon SNS 主題取得完成狀態。

前面圖所顯示的流程與分析文本和發票/收據的流程相同。您可以通過調用the section called “StartDocumentAnalysis”並開始分析發票/收據，方法是調用the section called “StartExpenseAnalysis”您可以通過調用the section called “GetDocumentAnalysis”或者the section called “GetExpenseAnalysis”分別。

啟動文字偵測

透過呼叫StartDocumentTextDetection。以下是由 `StartDocumentTextDetection` 傳遞的 JSON 請求範例。

```
{
    "DocumentLocation": {
        "S3Object": {
```

```
            "Bucket": "bucket",
            "Name": "image.pdf"
        }
    },
    "ClientRequestToken": "DocumentDetectionToken",
    "NotificationChannel": {
        "SNSTopicArn": "arn:aws:sns:us-east-1:nnnnnnnnnn:topic",
        "RoleArn": "arn:aws:iam::nnnnnnnnnn:role/roleTopic"
    },
    "JobTag": "Receipt"
}
```

輸入參數DocumentLocation提供檔案名稱以及從中取回檔案名稱的 Amazon S3 儲存貯體。NotificationChannel包含 Amazon SNS 主題的 Amazon Resource Name (ARN)，Amazon Amazon Textract N) 將在文本檢測請求完成時通知。Amazon SNS 主題必須與您呼叫的 Amazon Textract 端點位於同一個 AWS 區域。NotificationChannel也包含允許 Amazon Textract 發佈到 Amazon SNS 主題的角色 ARN。透過 IAM 服務角色，您將許可提供給 Amazon SNS 主題。如需詳細資訊，請參閱 為異步操作配置 Amazon Textract。

您也可以指定可選輸入參數JobTag，可讓您找出顯示完成狀態且已發佈至 Amazon SNS 主題的任務或任務組。例如，您可以使用JobTag標識正在處理的單據類型，例如納稅表或收據。

為避免分析任務發生意外的重複，您可以選擇性地提供等冪符記 ClientRequestToken。如果您提供ClientRequestToken，Start操作會傳回相同的JobId對於多個相同的調用Start操作，例如StartDocumentTextDetection。ClientRequestToken 符記有 7 天的存留期。在 7 天後，您可以再次使用它。如果您在符記的存留期內重新使用符記，會發生下列情況：

- 如果您使用的 Start 操作與相同的輸入參數來重新使用符記，將傳回相同的 JobId。不會再次執行任務，而 Amazon Textract 也不會傳送完成狀態到已註冊的 Amazon SNS 主題。
- 如果您以相同的 Start 操作搭配些微變更的參數來重新使用符記，您會得到一個 idempotentparametermismatchexception (HTTP 狀態碼：400) 例外狀況。
- 如果您以不同的 Start 操作來重新使用符記，操作將可成功。

另一個可選參數是OutputConfig，它允許您調整輸出的放置位置。默認情況下，Amazon Textract 將在內部存儲結果，並且只能通過獲取 API 操作訪問。搭配OutputConfig，您可以設置將輸出發送到的存儲桶的名稱以及結果的文件前綴，您可以在其中下載結果。此外，您也可以設定KMSKeyID參數添加到客戶託管密鑰來加密您的輸出。如果沒有此參數設置，Amazon Textract 將使用AWS 受管金鑰適用於 Amazon S3

> **Note**
>
> 在使用此參數之前，請確保您具有輸出存儲桶的 PutObject 權限。此外，請確保您具有解密、ReEncrypt、GenerateDataKey 和 DescribeKey 權限AWS KMS鍵，如果您決定使用它。

對於 StartDocumentTextDetection 操作的回應為任務識別碼 (JobId)。使用JobId追蹤請求並在 Amazon Textract 發佈完成狀態到 Amazon SNS 主題後取得分析結果。以下是範例：

```
{"JobId":"270c1cc5e1d0ea2fbc59d97cb69a72a5495da75851976b14a1784ca90fc180e3"}
```

如果同時啟動太多作業，請調用StartDocumentTextDetectionRETURNLimitExceededException例外狀況 (HTTP 狀態碼：400)，直到數量同時執行任務的數量低於 Amazon Textract 服務限制。

如果您發現限制拒絕例外狀況發生且產生活動激增，請考慮使用 Amazon SQS 排列來管理傳入的請求。聯絡AWS如果您 Support 現同時進行的請求的平均數量無法由 Amazon SQS 排列控管，而您仍收到LimitExceededException例外狀況。

取得 Amazon Textract 分析請求的完成狀態

Amazon Textract 將傳送分析完成通知道已註冊的 Amazon SNS 主題。通知包含任務識別碼和並以 JSON 字串顯示操作的完成狀態。成功的文本檢測請求具有SUCCEEDED狀態。例如，以下結果顯示文字偵測任務的成功處理。

```
{
    "JobId": "642492aea78a86a40665555dc375ee97bc963f342b29cd05030f19bd8fd1bc5f",
    "Status": "SUCCEEDED",
    "API": "StartDocumentTextDetection",
    "JobTag": "Receipt",
    "Timestamp": 1543599965969,
    "DocumentLocation": {
        "S3ObjectName": "document",
        "S3Bucket": "bucket"
    }
}
```

如需詳細資訊，請參閱 Amazon Textract 結果通知。

若要取得 Amazon Amazon Textract 發佈至 Amazon SNS 主題的狀態資訊，請使用以下其中一個選項：

- AWS Lambda— 您可以訂AWS Lambda函數，以寫入 Amazon SNS 主題。當 Amazon Textract 通知 Amazon SNS 主題請求已完成時，將呼叫此功能。如果您需要伺服器端程式碼來處理文字偵測請求的結果，請使用 Lambda 函數。例如，您可能想要使用服務器端代碼來標註映像，或在傳回信息到客户端應用程式前針對檢測到的文本創建報告。
- Amazon SQS— 您可以訂 Amazon SQS 佇列至 Amazon SNS 主題。接著請輪詢 Amazon SQS 排列以取得由 Amazon Textract 在文本檢測請求完成時發佈的完成狀態。如需詳細資訊，請參閱 檢測或分析多頁文檔中的文本。如果您只想從客户端應用程式呼叫 Amazon Textract 操作，請使用 Amazon SQS 佇列。

> **Important**
>
> 我們不建議您透過重複呼叫 Amazon Textract 來取得請求完成狀態Getoperation. 這是因為 Amazon Textract 會限制Get操作，如果執行太多請求。如果您同時處理多個檔案，監控一個 SQS 排列的完成通知將會比輪詢 Amazon Textract 以獲得個別任務的狀態更容易且有效率。

獲取 Amazon Textract 檢測結果

若要取得文本檢測請求結果，首先請確認自 Amazon SNS 主題檢索的完成狀態為SUCCEEDED。然後呼叫 `GetDocumentTextDetection`，將傳遞自 `StartDocumentTextDetection` 傳回的 `JobId` 值。請求 JSON 格式類似於以下範例：

```
{
    "JobId": "270c1cc5e1d0ea2fbc59d97cb69a72a5495da75851976b14a1784ca90fc180e3",
    "MaxResults": 10,
    "SortBy": "TIMESTAMP"
}
```

`JobId`是文本檢測操作的標識符。由於文字偵測可以生成大量數據，請使用`MaxResults`來指定傳回結果的最大數量，以在單個Getoperation. 的預設預設預設值`MaxResults`是 1,000 個。如果您指定的值大於 1,000，將只會傳回 1,000 個結果。如果操作未返回所有結果，將傳回下一頁的分頁令牌。若要取得下一頁的結果，請在`NextToken`參數。

> **Note**
>
> Amazon Textract 會保留異步操作的結果 7 天。在此時間後您無法取得結果。

所以此GetDocumentTextDetection操作響應 JSON 格式類似如下。檢測到的頁面總數在DocumentMetadata。檢測到的文本將在Blocks陣列。關於的資訊Block對象，請參閱文本檢測和文檔分析響應對象。

```
{
    "DocumentMetadata": {
        "Pages": 1
    },
    "JobStatus": "SUCCEEDED",
    "Blocks": [
        {
            "BlockType": "PAGE",
            "Geometry": {
                "BoundingBox": {
                    "Width": 1.0,
                    "Height": 1.0,
                    "Left": 0.0,
                    "Top": 0.0
                },
                "Polygon": [
                    {
                        "X": 0.0,
                        "Y": 0.0
                    },
                    {
                        "X": 1.0,
                        "Y": 0.0
                    },
                    {
                        "X": 1.0,
                        "Y": 1.0
                    },
                    {
                        "X": 0.0,
                        "Y": 1.0
                    }
                ]
```

```
            },
            "Id": "64533157-c47e-401a-930e-7ca1bb3ac3fa",
            "Relationships": [
                {
                    "Type": "CHILD",
                    "Ids": [
                        "4297834d-dcb1-413b-8908-3b96866ebbb5",
                        "1d85ba24-2877-4d09-b8b2-393833d769e9",
                        "193e9c47-fd87-475a-ba09-3fda210d8784",
                        "bd8aeb62-961b-4b47-b78a-e4ed9eeecd0f"
                    ]
                }
            ],
            "Page": 1
        },
        {
            "BlockType": "LINE",
            "Confidence": 53.301639556884766,
            "Text": "ellooworio",
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.9999999403953552,
                    "Height": 0.5365243554115295,
                    "Left": 0.0,
                    "Top": 0.46347561478614807
                },
                "Polygon": [
                    {
                        "X": 0.0,
                        "Y": 0.46347561478614807
                    },
                    {
                        "X": 0.9999999403953552,
                        "Y": 0.46347561478614807
                    },
                    {
                        "X": 0.9999999403953552,
                        "Y": 1.0
                    },
                    {
                        "X": 0.0,
                        "Y": 1.0
                    }
                ]
```

```
            },
            "Id": "4297834d-dcb1-413b-8908-3b96866ebbb5",
            "Relationships": [
                {
                    "Type": "CHILD",
                    "Ids": [
                        "170c3eb9-5155-4bec-8c44-173bba537e70"
                    ]
                }
            ],
            "Page": 1
        },
        {
            "BlockType": "LINE",
            "Confidence": 89.15632629394531,
            "Text": "He llo,",
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.33642634749412537,
                    "Height": 0.49159330129623413,
                    "Left": 0.13885067403316498,
                    "Top": 0.17169663310050964
                },
                "Polygon": [
                    {
                        "X": 0.13885067403316498,
                        "Y": 0.17169663310050964
                    },
                    {
                        "X": 0.47527703642845154,
                        "Y": 0.17169663310050964
                    },
                    {
                        "X": 0.47527703642845154,
                        "Y": 0.6632899641990662
                    },
                    {
                        "X": 0.13885067403316498,
                        "Y": 0.6632899641990662
                    }
                ]
            },
            "Id": "1d85ba24-2877-4d09-b8b2-393833d769e9",
            "Relationships": [
```

```
                {
                    "Type": "CHILD",
                    "Ids": [
                        "516ae823-3bab-4f9a-9d74-ad7150d128ab",
                        "6bcf4ea8-bbe8-4686-91be-b98dd63bc6a6"
                    ]
                }
            ],
            "Page": 1
        },
        {
            "BlockType": "LINE",
            "Confidence": 82.44834899902344,
            "Text": "worlo",
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.33182239532470703,
                    "Height": 0.3766750991344452,
                    "Left": 0.5091826915740967,
                    "Top": 0.23131252825260162
                },
                "Polygon": [
                    {
                        "X": 0.5091826915740967,
                        "Y": 0.23131252825260162
                    },
                    {
                        "X": 0.8410050868988037,
                        "Y": 0.23131252825260162
                    },
                    {
                        "X": 0.8410050868988037,
                        "Y": 0.607987642288208
                    },
                    {
                        "X": 0.5091826915740967,
                        "Y": 0.607987642288208
                    }
                ]
            },
            "Id": "193e9c47-fd87-475a-ba09-3fda210d8784",
            "Relationships": [
                {
                    "Type": "CHILD",
```

```
                    "Ids": [
                        "ed135c3b-35dd-4085-8f00-26aedab0125f"
                    ]
                }
            ],
            "Page": 1
        },
        {
            "BlockType": "LINE",
            "Confidence": 88.50325775146484,
            "Text": "world",
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.35004907846450806,
                    "Height": 0.19635874032974243,
                    "Left": 0.527581512928009,
                    "Top": 0.30100569128990173
                },
                "Polygon": [
                    {
                        "X": 0.527581512928009,
                        "Y": 0.30100569128990173
                    },
                    {
                        "X": 0.8776305913925171,
                        "Y": 0.30100569128990173
                    },
                    {
                        "X": 0.8776305913925171,
                        "Y": 0.49736443161964417
                    },
                    {
                        "X": 0.527581512928009,
                        "Y": 0.49736443161964417
                    }
                ]
            },
            "Id": "bd8aeb62-961b-4b47-b78a-e4ed9eeecd0f",
            "Relationships": [
                {
                    "Type": "CHILD",
                    "Ids": [
                        "9e28834d-798e-4a62-8862-a837dfd895a6"
                    ]
```

```
                }
            ],
            "Page": 1
        },
        {
            "BlockType": "WORD",
            "Confidence": 53.301639556884766,
            "Text": "ellooworio",
            "Geometry": {
                "BoundingBox": {
                    "Width": 1.0,
                    "Height": 0.5365243554115295,
                    "Left": 0.0,
                    "Top": 0.46347561478614807
                },
                "Polygon": [
                    {
                        "X": 0.0,
                        "Y": 0.46347561478614807
                    },
                    {
                        "X": 1.0,
                        "Y": 0.46347561478614807
                    },
                    {
                        "X": 1.0,
                        "Y": 1.0
                    },
                    {
                        "X": 0.0,
                        "Y": 1.0
                    }
                ]
            },
            "Id": "170c3eb9-5155-4bec-8c44-173bba537e70",
            "Page": 1
        },
        {
            "BlockType": "WORD",
            "Confidence": 88.46246337890625,
            "Text": "He",
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.15350718796253204,
```

```
                    "Height": 0.29955607652664185,
                    "Left": 0.13885067403316498,
                    "Top": 0.21856294572353363
                },
                "Polygon": [
                    {
                        "X": 0.13885067403316498,
                        "Y": 0.21856294572353363
                    },
                    {
                        "X": 0.292357861995697,
                        "Y": 0.21856294572353363
                    },
                    {
                        "X": 0.292357861995697,
                        "Y": 0.5181190371513367
                    },
                    {
                        "X": 0.13885067403316498,
                        "Y": 0.5181190371513367
                    }
                ]
            },
            "Id": "516ae823-3bab-4f9a-9d74-ad7150d128ab",
            "Page": 1
        },
        {
            "BlockType": "WORD",
            "Confidence": 89.8501968383789,
            "Text": "llo,",
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.17724157869815826,
                    "Height": 0.49159327149391174,
                    "Left": 0.2980354428291321,
                    "Top": 0.17169663310050964
                },
                "Polygon": [
                    {
                        "X": 0.2980354428291321,
                        "Y": 0.17169663310050964
                    },
                    {
                        "X": 0.47527703642845154,
```

```
                        "Y": 0.17169663310050964
                    },
                    {
                        "X": 0.47527703642845154,
                        "Y": 0.6632899045944214
                    },
                    {
                        "X": 0.2980354428291321,
                        "Y": 0.6632899045944214
                    }
                ]
            },
            "Id": "6bcf4ea8-bbe8-4686-91be-b98dd63bc6a6",
            "Page": 1
        },
        {
            "BlockType": "WORD",
            "Confidence": 82.44834899902344,
            "Text": "worlo",
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.33182239532470703,
                    "Height": 0.3766750991344452,
                    "Left": 0.5091826915740967,
                    "Top": 0.23131252825260162
                },
                "Polygon": [
                    {
                        "X": 0.5091826915740967,
                        "Y": 0.23131252825260162
                    },
                    {
                        "X": 0.8410050868988037,
                        "Y": 0.23131252825260162
                    },
                    {
                        "X": 0.8410050868988037,
                        "Y": 0.607987642288208
                    },
                    {
                        "X": 0.5091826915740967,
                        "Y": 0.607987642288208
                    }
                ]
```

```
            },
            "Id": "ed135c3b-35dd-4085-8f00-26aedab0125f",
            "Page": 1
        },
        {
            "BlockType": "WORD",
            "Confidence": 88.50325775146484,
            "Text": "world",
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.35004907846450806,
                    "Height": 0.19635874032974243,
                    "Left": 0.527581512928009,
                    "Top": 0.30100569128990173
                },
                "Polygon": [
                    {
                        "X": 0.527581512928009,
                        "Y": 0.30100569128990173
                    },
                    {
                        "X": 0.8776305913925171,
                        "Y": 0.30100569128990173
                    },
                    {
                        "X": 0.8776305913925171,
                        "Y": 0.49736443161964417
                    },
                    {
                        "X": 0.527581512928009,
                        "Y": 0.49736443161964417
                    }
                ]
            },
            "Id": "9e28834d-798e-4a62-8862-a837dfd895a6",
            "Page": 1
        }
    ]
}
```

為異步操作配置 Amazon Textract

以下程序介紹如何配置 Amazon Textract 以配合使用 Amazon Simple Notification Service (Amazon SNS) 主題和 Amazon Simple Queue Service (Amazon SQS) 排列。

> **Note**
>
> 如果您使用這些指示來設定檢測或分析多頁文檔中的文本例如，您不需要執行步驟 3 — 6。此範例包含用於建立和配置 Amazon SNS 主題和 Amazon SQS 佇列的程式碼。

若要配置 Amazon Textract

1. 設定AWS賬户以訪問 Amazon Textract。如需詳細資訊，請參閱 步驟 1：設定 AWS 帳户並建立 IAM 使用者。

   請確認用户至少有以下權限：

   - AmazonTextractFullAccess
   - AmazonS3ReadOnlyAccess
   - AmazonSNSFullAccess
   - AmazonSQSFullAccess
2. 安裝並設定所需的 AWS SDK。如需詳細資訊，請參閱 步驟 2：設定AWS CLI和AWS開發套件。
3. 建立 Amazon SNS 主題。將主題名稱前面加上卓越亞馬遜。請記下主題的 Amazon Resource Name (ARN)。請確定主題位於與AWS端點，以與 AWS 賬户一起使用。
4. 建立 Amazon SQS 標準隊列通過使用Amazon SQS 控制台。記下佇列 ARN。
5. 將佇列訂閱至您在步驟 3 中建立的主題。
6. 將許可提供給 Amazon SNS 主題，以將訊息傳送至 Amazon SQS 佇列。 。
7. 建立 IAM 服務角色，讓 Amazon Textract 存取 Amazon SNS 主題。記下服務角色的 Amazon Resource Name (ARN)。如需詳細資訊，請參閱 授予 Amazon Textract 對您的 Amazon SNS 主題的訪問權限。
8. 新增下列內嵌政策傳遞至您在步驟 1 中建立的 IAM 用户。

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Sid": "MySid",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "Service role ARN from step 7"
        }
    ]
}
```

將內嵌政策命名。

9. 您現在可以運行檢測或分析多頁文檔中的文本。

授予 Amazon Textract 對您的 Amazon SNS 主題的訪問權限

當異步操作完成時，Amazon Textract 需要獲得許可才能向您的 Amazon SNS 主題發送消息。您可以使用 IAM 服務角色，讓 Amazon Textract 存取 Amazon SNS 主題。

創建 Amazon SNS 主題時，必須在主題名稱前加上**AmazonTextract**— 例如，**AmazonTextractMyTopicName**。

1. 登入 IAM 主控台 (https://console.aws.amazon.com/iam)。
2. 在導覽窗格中，選擇 Roles (角色)。
3. 選擇 Create Role (建立角色)。
4. 對於 Select type of trusted entity (選取信任的實體類型)，選擇 AWS service (AWS 服務)。
5. 適用於選擇將使用此角色的服務，選擇Textract。
6. 選擇 Next: (下一步：) Permissions (許可)。
7. 驗證AmazonTextractServiceRole策略已包含在附加策略列表中。若要在清單中顯示政策，請在篩選政策。
8. 選擇 Next: (下一步：) Tags (標籤)。
9. 您不需要新增標籤，所以請選擇下一頁: Review (檢閱)。
10. 在 Review (檢閱) 區段中，針對 Role name (角色名稱)，輸入角色的名稱 (例如，`TextractRole`)。In角色描述，請更新該角色的描述，然後選擇建立角色。
11. 選擇新角色來開啟角色的詳細資訊頁面。
12. 在 Summary (摘要) 中，複製 Role ARN (角色 ARN) 值，並將其儲存。
13. 選擇 Trust relationships (信任關係)。

14. 選擇編輯信任關係，並確保信任策略如下所示。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "textract.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

15. 選擇 Update Trust Policy (更新信任政策)。

檢測或分析多頁文檔中的文本

此過程介紹如何使用 Amazon Amazon Textract 檢測操作、存儲在 Amazon S3 存儲桶中的文檔、Amazon SNS 主題和 Amazon SQS 隊列來檢測或分析多頁文檔中的文本。多頁文件處理是一種非同步操作。如需詳細資訊，請參閱 調用 Amazon Textract 異步操作。

您可以選擇希望代碼執行的處理類型：文本檢測、文本分析或費用分析。

處理結果返回在the section called “Block”對象，這些對象會根據您使用的處理類型而定。

要檢測多頁文檔中的文本或分析，請執行以下操作：

1. 建立 Amazon SNS 主題和 Amazon SQS 佇列。
2. 訂佇列。
3. 將許可提供給主題，以將訊息傳送至佇列。
4. 開始處理文檔。為您選擇的分析類型使用適當的操作：
   - StartDocumentTextDetection用於文本檢測任務。
   - StartDocumentAnalysis用於文本分析任務。
   - StartExpenseAnalysis用於支出分析任務。
5. 從 Amazon SQS 佇列取得完成狀態。示例代碼跟蹤作業標識符 (`JobId`)，這是由`Startoperation`. 它只會顯示從完成狀態中讀取後，只會顯示相符的工作識別碼相符的結果。如果其他應用程式使用

相同的佇列和主題，這麼做非常重要。為了方便起見，此範例刪除不符合的任務。考慮將已刪除的任務添加到 Amazon SQS 無效字母佇列以供進一步調查。

6. 通過調用所選分析類型的相應操作來獲取並顯示處理結果：
   - GetDocumentTextDetection用於文本檢測任務。
   - GetDocumentAnalysis用於文本分析任務。
   - GetExpenseAnalysis用於支出分析任務。
7. 刪除 Amazon SNS 主題和 Amazon SQS 佇列。

執行異步操作

此程序的範例程式碼以 Java、Python 提供，AWS CLI。開始之前，請先安裝適當的AWS開發套件。如需詳細資訊，請參閱 步驟 2：設定AWS CLI和AWS開發套件。

檢測或分析多頁文檔中的文本

1. 請配置用户對 Amazon Textract 的訪問，以及配置 Amazon Textract 存取 Amazon SNS 的 Amazon Textract 存取。如需詳細資訊，請參閱 為異步操作配置 Amazon Textract。若要完成此程序，您需要 PDF 格式的多頁檔案。請跳過步驟 3 — 6，因為範例程式碼會建立並設定 Amazon SNS 主題和 Amazon SQS 佇列。如果壓縮t在 CLI 範例中，您不需要設定 SQS 佇列。
2. 將 PDF 或 TIFF 格式的多頁文件上傳至 Amazon S3 儲存貯體。（也可以處理 JPEG、PNG、TIFF 或 PDF 格式的單頁文檔）。

   如需說明，請參閱「」將對象上傳至 Amazon S3中的Amazon Simple Storage Service 用户指南。
3. 使用下列內容AWS SDK for Java、SDK for Python (Boto3) 或AWS CLI代碼來檢測多頁文檔中的文本或分析文本。在 中`main`函數：
   - 替換`roleArn`使用您保存在的 IAM 角色 ARN授予 Amazon Textract 對您的 Amazon SNS 主題的訪問權限。
   - 替換`bucket`和`document`以您在步驟 2 中指定的儲存貯體與影片檔名稱。
   - 取代`type`的輸入參數`ProcessDocument`函數與您想執行的處理類型。使用`ProcessType.DETECTION`來檢測文本。使用`ProcessType.ANALYSIS`來分析文本。
   - 對於 Python 示例，將`region_name`與您的客户端所在的區域一起運行。

   對於AWS CLI例如，請執行以下操作：

- 在調用StartDocumentTextDetection中，替換bucket-name以您的 S3 儲存貯體名稱取代file-name以您在步驟 2 中指定的檔案名稱來取代。通過替換region-name與您所在地區的名稱相同。請注意，CLI 示例沒有使用 SQS。
- 在調用GetDocumentTextDetection更換job-id-number使用job-id返回的StartDocumentTextDetection。通過替換region-name與您所在地區的名稱相同。

Java

```
package com.amazonaws.samples;

import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import com.amazonaws.auth.policy.Condition;
import com.amazonaws.auth.policy.Policy;
import com.amazonaws.auth.policy.Principal;
import com.amazonaws.auth.policy.Resource;
import com.amazonaws.auth.policy.Statement;
import com.amazonaws.auth.policy.Statement.Effect;
import com.amazonaws.auth.policy.actions.SQSActions;
import com.amazonaws.services.sns.AmazonSNS;
import com.amazonaws.services.sns.AmazonSNSClientBuilder;
import com.amazonaws.services.sns.model.CreateTopicRequest;
import com.amazonaws.services.sns.model.CreateTopicResult;
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.CreateQueueRequest;
import com.amazonaws.services.sqs.model.Message;
import com.amazonaws.services.sqs.model.QueueAttributeName;
import com.amazonaws.services.sqs.model.SetQueueAttributesRequest;
import com.amazonaws.services.textract.AmazonTextract;
import com.amazonaws.services.textract.AmazonTextractClientBuilder;
import com.amazonaws.services.textract.model.Block;
import com.amazonaws.services.textract.model.DocumentLocation;
import com.amazonaws.services.textract.model.DocumentMetadata;
import com.amazonaws.services.textract.model.GetDocumentAnalysisRequest;
import com.amazonaws.services.textract.model.GetDocumentAnalysisResult;
import com.amazonaws.services.textract.model.GetDocumentTextDetectionRequest;
import com.amazonaws.services.textract.model.GetDocumentTextDetectionResult;
```

```
import com.amazonaws.services.textract.model.NotificationChannel;
import com.amazonaws.services.textract.model.Relationship;
import com.amazonaws.services.textract.model.S3Object;
import com.amazonaws.services.textract.model.StartDocumentAnalysisRequest;
import com.amazonaws.services.textract.model.StartDocumentAnalysisResult;
import com.amazonaws.services.textract.model.StartDocumentTextDetectionRequest;
import com.amazonaws.services.textract.model.StartDocumentTextDetectionResult;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;;
public class DocumentProcessor {

    private static String sqsQueueName=null;
    private static String snsTopicName=null;
    private static String snsTopicArn = null;
    private static String roleArn= null;
    private static String sqsQueueUrl = null;
    private static String sqsQueueArn = null;
    private static String startJobId = null;
    private static String bucket = null;
    private static String document = null;
    private static AmazonSQS sqs=null;
    private static AmazonSNS sns=null;
    private static AmazonTextract textract = null;

    public enum ProcessType {
        DETECTION,ANALYSIS
    }

    public static void main(String[] args) throws Exception {

        String document = "document";
        String bucket = "bucket";
        String roleArn="role";

        sns = AmazonSNSClientBuilder.defaultClient();
        sqs= AmazonSQSClientBuilder.defaultClient();
        textract=AmazonTextractClientBuilder.defaultClient();

        CreateTopicandQueue();
        ProcessDocument(bucket,document,roleArn,ProcessType.DETECTION);
        DeleteTopicandQueue();
        System.out.println("Done!");
```

```
   }
   // Creates an SNS topic and SQS queue. The queue is subscribed to the
topic.
   static void CreateTopicandQueue()
   {
       //create a new SNS topic
       snsTopicName="AmazonTextractTopic" +
Long.toString(System.currentTimeMillis());
       CreateTopicRequest createTopicRequest = new
CreateTopicRequest(snsTopicName);
       CreateTopicResult createTopicResult =
sns.createTopic(createTopicRequest);
       snsTopicArn=createTopicResult.getTopicArn();

       //Create a new SQS Queue
       sqsQueueName="AmazonTextractQueue" +
Long.toString(System.currentTimeMillis());
       final CreateQueueRequest createQueueRequest = new
CreateQueueRequest(sqsQueueName);
       sqsQueueUrl = sqs.createQueue(createQueueRequest).getQueueUrl();
       sqsQueueArn = sqs.getQueueAttributes(sqsQueueUrl,
Arrays.asList("QueueArn")).getAttributes().get("QueueArn");

       //Subscribe SQS queue to SNS topic
       String sqsSubscriptionArn = sns.subscribe(snsTopicArn, "sqs",
sqsQueueArn).getSubscriptionArn();

       // Authorize queue
         Policy policy = new Policy().withStatements(
                 new Statement(Effect.Allow)
                 .withPrincipals(Principal.AllUsers)
                 .withActions(SQSActions.SendMessage)
                 .withResources(new Resource(sqsQueueArn))
                 .withConditions(new
Condition().withType("ArnEquals").withConditionKey("aws:SourceArn").withValues(snsTopic
                 );


         Map queueAttributes = new HashMap();
         queueAttributes.put(QueueAttributeName.Policy.toString(),
policy.toJson());
         sqs.setQueueAttributes(new SetQueueAttributesRequest(sqsQueueUrl,
queueAttributes));
```

```
         System.out.println("Topic arn: " + snsTopicArn);
         System.out.println("Queue arn: " + sqsQueueArn);
         System.out.println("Queue url: " + sqsQueueUrl);
         System.out.println("Queue sub arn: " + sqsSubscriptionArn );
     }
   static void DeleteTopicandQueue()
   {
       if (sqs !=null) {
           sqs.deleteQueue(sqsQueueUrl);
           System.out.println("SQS queue deleted");
       }

       if (sns!=null) {
           sns.deleteTopic(snsTopicArn);
           System.out.println("SNS topic deleted");
       }
   }

   //Starts the processing of the input document.
   static void ProcessDocument(String inBucket, String inDocument, String
inRoleArn, ProcessType type) throws Exception
   {
       bucket=inBucket;
       document=inDocument;
       roleArn=inRoleArn;

       switch(type)
       {
           case DETECTION:
               StartDocumentTextDetection(bucket, document);
               System.out.println("Processing type: Detection");
               break;
           case ANALYSIS:
               StartDocumentAnalysis(bucket,document);
               System.out.println("Processing type: Analysis");
               break;
           default:
               System.out.println("Invalid processing type. Choose Detection or
Analysis");
               throw new Exception("Invalid processing type");

       }
```

```
        System.out.println("Waiting for job: " + startJobId);
        //Poll queue for messages
        List<Message> messages=null;
        int dotLine=0;
        boolean jobFound=false;

        //loop until the job status is published. Ignore other messages in
queue.
        do{
            messages = sqs.receiveMessage(sqsQueueUrl).getMessages();
            if (dotLine++<40){
                System.out.print(".");
            }else{
                System.out.println();
                dotLine=0;
            }

            if (!messages.isEmpty()) {
                //Loop through messages received.
                for (Message message: messages) {
                    String notification = message.getBody();

                    // Get status and job id from notification.
                    ObjectMapper mapper = new ObjectMapper();
                    JsonNode jsonMessageTree = mapper.readTree(notification);
                    JsonNode messageBodyText = jsonMessageTree.get("Message");
                    ObjectMapper operationResultMapper = new ObjectMapper();
                    JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
                    JsonNode operationJobId = jsonResultTree.get("JobId");
                    JsonNode operationStatus = jsonResultTree.get("Status");
                    System.out.println("Job found was " + operationJobId);
                    // Found job. Get the results and display.
                    if(operationJobId.asText().equals(startJobId)){
                        jobFound=true;
                        System.out.println("Job id: " + operationJobId );
                        System.out.println("Status : " +
operationStatus.toString());
                        if (operationStatus.asText().equals("SUCCEEDED")){
                            switch(type)
                            {
                                case DETECTION:
                                    GetDocumentTextDetectionResults();
                                    break;
```

```
                                case ANALYSIS:
                                    GetDocumentAnalysisResults();
                                    break;
                                default:
                                    System.out.println("Invalid processing type.
Choose Detection or Analysis");
                                    throw new Exception("Invalid processing
type");

                            }
                        }
                        else{
                            System.out.println("Document analysis failed");
                        }


sqs.deleteMessage(sqsQueueUrl,message.getReceiptHandle());
                    }

                    else{
                        System.out.println("Job received was not job " +
startJobId);
                        //Delete unknown message. Consider moving message to
dead letter queue

sqs.deleteMessage(sqsQueueUrl,message.getReceiptHandle());
                    }
                }
            }
            else {
                Thread.sleep(5000);
            }
        } while (!jobFound);

        System.out.println("Finished processing document");
    }

    private static void StartDocumentTextDetection(String bucket, String
document) throws Exception{

        //Create notification channel
        NotificationChannel channel= new NotificationChannel()
                .withSNSTopicArn(snsTopicArn)
                .withRoleArn(roleArn);
```

```
       StartDocumentTextDetectionRequest req = new
StartDocumentTextDetectionRequest()
               .withDocumentLocation(new DocumentLocation()
                   .withS3Object(new S3Object()
                       .withBucket(bucket)
                       .withName(document)))
               .withJobTag("DetectingText")
               .withNotificationChannel(channel);

       StartDocumentTextDetectionResult startDocumentTextDetectionResult =
textract.startDocumentTextDetection(req);
       startJobId=startDocumentTextDetectionResult.getJobId();
   }

 //Gets the results of processing started by StartDocumentTextDetection
   private static void GetDocumentTextDetectionResults() throws Exception{
       int maxResults=1000;
       String paginationToken=null;
       GetDocumentTextDetectionResult response=null;
       Boolean finished=false;

       while (finished==false)
       {
           GetDocumentTextDetectionRequest documentTextDetectionRequest= new
GetDocumentTextDetectionRequest()
                   .withJobId(startJobId)
                   .withMaxResults(maxResults)
                   .withNextToken(paginationToken);
           response =
textract.getDocumentTextDetection(documentTextDetectionRequest);
           DocumentMetadata documentMetaData=response.getDocumentMetadata();

           System.out.println("Pages: " +
documentMetaData.getPages().toString());

           //Show blocks information
           List<Block> blocks= response.getBlocks();
           for (Block block : blocks) {
               DisplayBlockInfo(block);
           }
           paginationToken=response.getNextToken();
           if (paginationToken==null)
               finished=true;
```

```
        }

    }

    private static void StartDocumentAnalysis(String bucket, String document)
throws Exception{
        //Create notification channel
        NotificationChannel channel= new NotificationChannel()
                .withSNSTopicArn(snsTopicArn)
                .withRoleArn(roleArn);

        StartDocumentAnalysisRequest req = new StartDocumentAnalysisRequest()
                .withFeatureTypes("TABLES","FORMS")
                .withDocumentLocation(new DocumentLocation()
                    .withS3Object(new S3Object()
                        .withBucket(bucket)
                        .withName(document)))
                .withJobTag("AnalyzingText")
                .withNotificationChannel(channel);

        StartDocumentAnalysisResult startDocumentAnalysisResult =
textract.startDocumentAnalysis(req);
        startJobId=startDocumentAnalysisResult.getJobId();
    }
    //Gets the results of processing started by StartDocumentAnalysis
    private static void GetDocumentAnalysisResults() throws Exception{

        int maxResults=1000;
        String paginationToken=null;
        GetDocumentAnalysisResult response=null;
        Boolean finished=false;

        //loops until pagination token is null
        while (finished==false)
        {
            GetDocumentAnalysisRequest documentAnalysisRequest= new
GetDocumentAnalysisRequest()
                    .withJobId(startJobId)
                    .withMaxResults(maxResults)
                    .withNextToken(paginationToken);

            response = textract.getDocumentAnalysis(documentAnalysisRequest);
```

```
            DocumentMetadata documentMetaData=response.getDocumentMetadata();

            System.out.println("Pages: " +
documentMetaData.getPages().toString());

            //Show blocks, confidence and detection times
            List<Block> blocks= response.getBlocks();

            for (Block block : blocks) {
                DisplayBlockInfo(block);
            }
            paginationToken=response.getNextToken();
            if (paginationToken==null)
                finished=true;
        }

    }
    //Displays Block information for text detection and text analysis
    private static void DisplayBlockInfo(Block block) {
        System.out.println("Block Id : " + block.getId());
        if (block.getText()!=null)
            System.out.println("\tDetected text: " + block.getText());
        System.out.println("\tType: " + block.getBlockType());

        if (block.getBlockType().equals("PAGE") !=true) {
            System.out.println("\tConfidence: " +
block.getConfidence().toString());
        }
        if(block.getBlockType().equals("CELL"))
        {
            System.out.println("\tCell information:");
            System.out.println("\t\tColumn: " + block.getColumnIndex());
            System.out.println("\t\tRow: " + block.getRowIndex());
            System.out.println("\t\tColumn span: " + block.getColumnSpan());
            System.out.println("\t\tRow span: " + block.getRowSpan());

        }

        System.out.println("\tRelationships");
        List<Relationship> relationships=block.getRelationships();
        if(relationships!=null) {
            for (Relationship relationship : relationships) {
                System.out.println("\t\tType: " + relationship.getType());
```

```
                System.out.println("\t\tIDs: " +
 relationship.getIds().toString());
            }
        } else {
            System.out.println("\t\tNo related Blocks");
        }

        System.out.println("\tGeometry");
        System.out.println("\t\tBounding Box: " +
 block.getGeometry().getBoundingBox().toString());
        System.out.println("\t\tPolygon: " +
 block.getGeometry().getPolygon().toString());

        List<String> entityTypes = block.getEntityTypes();

        System.out.println("\tEntity Types");
        if(entityTypes!=null) {
            for (String entityType : entityTypes) {
                System.out.println("\t\tEntity Type: " + entityType);
            }
        } else {
            System.out.println("\t\tNo entity type");
        }

        if(block.getBlockType().equals("SELECTION_ELEMENT")) {
            System.out.print("    Selection element detected: ");
            if (block.getSelectionStatus().equals("SELECTED")){
                System.out.println("Selected");
            }else {
                System.out.println(" Not selected");
            }
        }
        if(block.getPage()!=null)
            System.out.println("\tPage: " + block.getPage());
        System.out.println();
    }
}
```

AWS CLI

這一個AWS CLI命令會開始非同步偵測指定文檔中的文字。其會傳回job-id來取代偵測結果。

```
aws textract start-document-text-detection --document-location
"{\"S3Object\":{\"Bucket\":\"bucket-name\",\"Name\":\"file-name\"}}" --
region region-name
```

這一個AWS CLI命令返回 Amazon Textract 異步操作的結果，如果提供job-id。

```
aws textract get-document-text-detection --region region-name --job-id job-id-
number
```

如果您正在 Windows 設備上訪問 CLI，請使用雙引號而不是單引號，並用反斜槓（即\）轉義內部雙引號以解決您可能遇到的任何解析器錯誤。例如，請參下列內容

```
aws textract start-document-text-detection --document-location "{\"S3Object\":
{\"Bucket\":\"bucket\",\"Name\":\"document\"}}" --region region-name
```

Python

```
import boto3
import json
import sys
import time


class ProcessType:
    DETECTION = 1
    ANALYSIS = 2


class DocumentProcessor:
    jobId = ''
    region_name = ''

    roleArn = ''
    bucket = ''
    document = ''

    sqsQueueUrl = ''
    snsTopicArn = ''
    processType = ''
```

```
    def __init__(self, role, bucket, document, region):
        self.roleArn = role
        self.bucket = bucket
        self.document = document
        self.region_name = region

        self.textract = boto3.client('textract', region_name=self.region_name)
        self.sqs = boto3.client('sqs')
        self.sns = boto3.client('sns')

    def ProcessDocument(self, type):
        jobFound = False

        self.processType = type
        validType = False

        # Determine which type of processing to perform
        if self.processType == ProcessType.DETECTION:
            response = self.textract.start_document_text_detection(
                DocumentLocation={'S3Object': {'Bucket': self.bucket, 'Name':
self.document}},
                NotificationChannel={'RoleArn': self.roleArn, 'SNSTopicArn':
self.snsTopicArn})
            print('Processing type: Detection')
            validType = True

        if self.processType == ProcessType.ANALYSIS:
            response = self.textract.start_document_analysis(
                DocumentLocation={'S3Object': {'Bucket': self.bucket, 'Name':
self.document}},
                FeatureTypes=["TABLES", "FORMS"],
                NotificationChannel={'RoleArn': self.roleArn, 'SNSTopicArn':
self.snsTopicArn})
            print('Processing type: Analysis')
            validType = True

        if validType == False:
            print("Invalid processing type. Choose Detection or Analysis.")
            return

        print('Start Job Id: ' + response['JobId'])
        dotLine = 0
        while jobFound == False:
```

```
            sqsResponse = self.sqs.receive_message(QueueUrl=self.sqsQueueUrl,
MessageAttributeNames=['ALL'],
                                                   MaxNumberOfMessages=10)

            if sqsResponse:

                if 'Messages' not in sqsResponse:
                    if dotLine < 40:
                        print('.', end='')
                        dotLine = dotLine + 1
                    else:
                        print()
                        dotLine = 0
                    sys.stdout.flush()
                    time.sleep(5)
                    continue

                for message in sqsResponse['Messages']:
                    notification = json.loads(message['Body'])
                    textMessage = json.loads(notification['Message'])
                    print(textMessage['JobId'])
                    print(textMessage['Status'])
                    if str(textMessage['JobId']) == response['JobId']:
                        print('Matching Job Found:' + textMessage['JobId'])
                        jobFound = True
                        self.GetResults(textMessage['JobId'])
                        self.sqs.delete_message(QueueUrl=self.sqsQueueUrl,

ReceiptHandle=message['ReceiptHandle'])
                    else:
                        print("Job didn't match:" +
                              str(textMessage['JobId']) + ' : ' +
str(response['JobId']))
                    # Delete the unknown message. Consider sending to dead
letter queue
                    self.sqs.delete_message(QueueUrl=self.sqsQueueUrl,

ReceiptHandle=message['ReceiptHandle'])

        print('Done!')

    def CreateTopicandQueue(self):

        millis = str(int(round(time.time() * 1000)))
```

```
        # Create SNS topic
        snsTopicName = "AmazonTextractTopic" + millis

        topicResponse = self.sns.create_topic(Name=snsTopicName)
        self.snsTopicArn = topicResponse['TopicArn']

        # create SQS queue
        sqsQueueName = "AmazonTextractQueue" + millis
        self.sqs.create_queue(QueueName=sqsQueueName)
        self.sqsQueueUrl = self.sqs.get_queue_url(QueueName=sqsQueueName)
['QueueUrl']

        attribs = self.sqs.get_queue_attributes(QueueUrl=self.sqsQueueUrl,
                                                AttributeNames=['QueueArn'])
['Attributes']

        sqsQueueArn = attribs['QueueArn']

        # Subscribe SQS queue to SNS topic
        self.sns.subscribe(
            TopicArn=self.snsTopicArn,
            Protocol='sqs',
            Endpoint=sqsQueueArn)

        # Authorize SNS to write SQS queue
        policy = """{{
  "Version":"2012-10-17",
  "Statement":[
    {{
      "Sid":"MyPolicy",
      "Effect":"Allow",
      "Principal" : {{"AWS" : "*"}},
      "Action":"SQS:SendMessage",
      "Resource": "{}",
      "Condition":{{
        "ArnEquals":{{
          "aws:SourceArn": "{}"
        }}
      }}
    }}
  ]
}}""".format(sqsQueueArn, self.snsTopicArn)
```

```
        response = self.sqs.set_queue_attributes(
            QueueUrl=self.sqsQueueUrl,
            Attributes={
                'Policy': policy
            })

    def DeleteTopicandQueue(self):
        self.sqs.delete_queue(QueueUrl=self.sqsQueueUrl)
        self.sns.delete_topic(TopicArn=self.snsTopicArn)

    # Display information about a block
    def DisplayBlockInfo(self, block):

        print("Block Id: " + block['Id'])
        print("Type: " + block['BlockType'])
        if 'EntityTypes' in block:
            print('EntityTypes: {}'.format(block['EntityTypes']))

        if 'Text' in block:
            print("Text: " + block['Text'])

        if block['BlockType'] != 'PAGE':
            print("Confidence: " + "{:.2f}".format(block['Confidence']) + "%")

        print('Page: {}'.format(block['Page']))

        if block['BlockType'] == 'CELL':
            print('Cell Information')
            print('\tColumn: {} '.format(block['ColumnIndex']))
            print('\tRow: {}'.format(block['RowIndex']))
            print('\tColumn span: {} '.format(block['ColumnSpan']))
            print('\tRow span: {}'.format(block['RowSpan']))

            if 'Relationships' in block:
                print('\tRelationships: {}'.format(block['Relationships']))

        print('Geometry')
        print('\tBounding Box: {}'.format(block['Geometry']['BoundingBox']))
        print('\tPolygon: {}'.format(block['Geometry']['Polygon']))

        if block['BlockType'] == 'SELECTION_ELEMENT':
            print('    Selection element detected: ', end='')
            if block['SelectionStatus'] == 'SELECTED':
                print('Selected')
```

```
            else:
                print('Not selected')

    def GetResults(self, jobId):
        maxResults = 1000
        paginationToken = None
        finished = False

        while finished == False:

            response = None

            if self.processType == ProcessType.ANALYSIS:
                if paginationToken == None:
                    response = self.textract.get_document_analysis(JobId=jobId,

MaxResults=maxResults)
                else:
                    response = self.textract.get_document_analysis(JobId=jobId,

MaxResults=maxResults,

NextToken=paginationToken)

            if self.processType == ProcessType.DETECTION:
                if paginationToken == None:
                    response =
self.textract.get_document_text_detection(JobId=jobId,

MaxResults=maxResults)
                else:
                    response =
self.textract.get_document_text_detection(JobId=jobId,

MaxResults=maxResults,

NextToken=paginationToken)

            blocks = response['Blocks']
            print('Detected Document Text')
            print('Pages: {}'.format(response['DocumentMetadata']['Pages']))

            # Display block information
            for block in blocks:
```

```
                self.DisplayBlockInfo(block)
                print()
                print()

            if 'NextToken' in response:
                paginationToken = response['NextToken']
            else:
                finished = True

    def GetResultsDocumentAnalysis(self, jobId):
        maxResults = 1000
        paginationToken = None
        finished = False

        while finished == False:

            response = None
            if paginationToken == None:
                response = self.textract.get_document_analysis(JobId=jobId,

MaxResults=maxResults)
            else:
                response = self.textract.get_document_analysis(JobId=jobId,

MaxResults=maxResults,

NextToken=paginationToken)

                # Get the text blocks
            blocks = response['Blocks']
            print('Analyzed Document Text')
            print('Pages: {}'.format(response['DocumentMetadata']['Pages']))
            # Display block information
            for block in blocks:
                self.DisplayBlockInfo(block)
                print()
                print()

                if 'NextToken' in response:
                    paginationToken = response['NextToken']
                else:
                    finished = True
```

```
def main():
    roleArn = ''
    bucket = ''
    document = ''
    region_name = ''

    analyzer = DocumentProcessor(roleArn, bucket, document, region_name)
    analyzer.CreateTopicandQueue()
    analyzer.ProcessDocument(ProcessType.DETECTION)
    analyzer.DeleteTopicandQueue()


if __name__ == "__main__":
    main()
```

Node.JS

在本範例中，將roleArn使用您保存在的 IAM 角色 ARN授予 Amazon Textract 對您的 Amazon SNS 主題的訪問權限。替換bucket和document以您在上述步驟 2 中指定的儲存貯體與影片檔名稱。替換processType與您想要在輸入文檔上使用的處理類型。最後，將REGION與您的客户端所在的區域一起運行。

```
 // snippet-start:[sqs.JavaScript.queues.createQueueV3]
// Import required AWS SDK clients and commands for Node.js
import { CreateQueueCommand, GetQueueAttributesCommand, GetQueueUrlCommand,
    SetQueueAttributesCommand, DeleteQueueCommand, ReceiveMessageCommand,
 DeleteMessageCommand } from  "@aws-sdk/client-sqs";
  import {CreateTopicCommand, SubscribeCommand, DeleteTopicCommand } from "@aws-
sdk/client-sns";
  import  { SQSClient } from "@aws-sdk/client-sqs";
  import  { SNSClient } from "@aws-sdk/client-sns";
  import  { TextractClient, StartDocumentTextDetectionCommand,
 StartDocumentAnalysisCommand, GetDocumentAnalysisCommand,
 GetDocumentTextDetectionCommand, DocumentMetadata } from "@aws-sdk/client-
textract";
  import { stdout } from "process";

  // Set the AWS Region.
  const REGION = "us-east-1"; //e.g. "us-east-1"
  // Create SNS service object.
  const sqsClient = new SQSClient({ region: REGION });
  const snsClient = new SNSClient({ region: REGION });
  const textractClient = new TextractClient({ region: REGION });
```

```
 // Set bucket and video variables
 const bucket = "bucket-name";

 const documentName = "document-name";
 const roleArn = "role-arn"
 const processType = "DETECTION"
 var startJobId = ""

 var ts = Date.now();
 const snsTopicName = "AmazonTextractExample" + ts;
 const snsTopicParams = {Name: snsTopicName}
 const sqsQueueName = "AmazonTextractQueue-" + ts;

 // Set the parameters
 const sqsParams = {
   QueueName: sqsQueueName, //SQS_QUEUE_URL
   Attributes: {
     DelaySeconds: "60", // Number of seconds delay.
     MessageRetentionPeriod: "86400", // Number of seconds delay.
   },
 };

 // Process a document based on operation type
 const processDocumment = async (type, bucket, videoName, roleArn, sqsQueueUrl,
snsTopicArn) =>
   {
   try
   {
       // Set job found and success status to false initially
     var jobFound = false
     var succeeded = false
     var dotLine = 0
     var processType = type
     var validType = false

     if (processType == "DETECTION"){
       var response = await textractClient.send(new
StartDocumentTextDetectionCommand({DocumentLocation:{S3Object:{Bucket:bucket,
Name:videoName}},
         NotificationChannel:{RoleArn: roleArn, SNSTopicArn: snsTopicArn}}))
       console.log("Processing type: Detection")
       validType = true
     }
```

```
     if (processType == "ANALYSIS"){
       var response = await textractClient.send(new
StartDocumentAnalysisCommand({DocumentLocation:{S3Object:{Bucket:bucket,
Name:videoName}},
         NotificationChannel:{RoleArn: roleArn, SNSTopicArn: snsTopicArn}}))
       console.log("Processing type: Analysis")
       validType = true
     }

     if (validType == false){
         console.log("Invalid processing type. Choose Detection or Analysis.")
         return
     }
   // while not found, continue to poll for response
   console.log(`Start Job ID: ${response.JobId}`)
   while (jobFound == false){
     var sqsReceivedResponse = await sqsClient.send(new
ReceiveMessageCommand({QueueUrl:sqsQueueUrl,
       MaxNumberOfMessages:'ALL', MaxNumberOfMessages:10}));
     if (sqsReceivedResponse){
       var responseString = JSON.stringify(sqsReceivedResponse)
       if (!responseString.includes('Body')){
         if (dotLine < 40) {
           console.log('.')
           dotLine = dotLine + 1
         }else {
           console.log('')
           dotLine = 0
         };
         stdout.write('', () => {
           console.log('');
         });
         await new Promise(resolve => setTimeout(resolve, 5000));
         continue
       }
     }

       // Once job found, log Job ID and return true if status is succeeded
       for (var message of sqsReceivedResponse.Messages){
           console.log("Retrieved messages:")
           var notification = JSON.parse(message.Body)
           var rekMessage = JSON.parse(notification.Message)
           var messageJobId = rekMessage.JobId
```

```
            if (String(rekMessage.JobId).includes(String(startJobId))){
                console.log('Matching job found:')
                console.log(rekMessage.JobId)
                jobFound = true
                // GET RESUlTS FUNCTION HERE
                var operationResults = await GetResults(processType,
rekMessage.JobId)
                //GET RESULTS FUMCTION HERE
                console.log(rekMessage.Status)
            if (String(rekMessage.Status).includes(String("SUCCEEDED"))){
                succeeded = true
                console.log("Job processing succeeded.")
                var sqsDeleteMessage = await sqsClient.send(new
DeleteMessageCommand({QueueUrl:sqsQueueUrl,
ReceiptHandle:message.ReceiptHandle}));
            }
            }else{
            console.log("Provided Job ID did not match returned ID.")
            var sqsDeleteMessage = await sqsClient.send(new
DeleteMessageCommand({QueueUrl:sqsQueueUrl,
ReceiptHandle:message.ReceiptHandle}));
            }
       }

   console.log("Done!")
   }
   }catch (err) {
       console.log("Error", err);
     }
 }

 // Create the SNS topic and SQS Queue
 const createTopicandQueue = async () => {
   try {
     // Create SNS topic
     const topicResponse = await snsClient.send(new
CreateTopicCommand(snsTopicParams));
     const topicArn = topicResponse.TopicArn
     console.log("Success", topicResponse);
     // Create SQS Queue
     const sqsResponse = await sqsClient.send(new
CreateQueueCommand(sqsParams));
     console.log("Success", sqsResponse);
```

```
     const sqsQueueCommand = await sqsClient.send(new
GetQueueUrlCommand({QueueName: sqsQueueName}))
     const sqsQueueUrl = sqsQueueCommand.QueueUrl
     const attribsResponse = await sqsClient.send(new
GetQueueAttributesCommand({QueueUrl: sqsQueueUrl, AttributeNames:
['QueueArn']}))
     const attribs = attribsResponse.Attributes
     console.log(attribs)
     const queueArn = attribs.QueueArn
     // subscribe SQS queue to SNS topic
     const subscribed = await snsClient.send(new SubscribeCommand({TopicArn:
topicArn, Protocol:'sqs', Endpoint: queueArn}))
     const policy = {
       Version: "2012-10-17",
       Statement: [
         {
           Sid: "MyPolicy",
           Effect: "Allow",
           Principal: {AWS: "*"},
           Action: "SQS:SendMessage",
           Resource: queueArn,
           Condition: {
             ArnEquals: {
               'aws:SourceArn': topicArn
             }
           }
         }
       ]
     };

     const response = sqsClient.send(new SetQueueAttributesCommand({QueueUrl:
sqsQueueUrl, Attributes: {Policy: JSON.stringify(policy)}}))
     console.log(response)
     console.log(sqsQueueUrl, topicArn)
     return [sqsQueueUrl, topicArn]

   } catch (err) {
     console.log("Error", err);

   }
 }

 const deleteTopicAndQueue = async (sqsQueueUrlArg, snsTopicArnArg) => {
```

```
    const deleteQueue = await sqsClient.send(new DeleteQueueCommand({QueueUrl:
sqsQueueUrlArg}));
    const deleteTopic = await snsClient.send(new DeleteTopicCommand({TopicArn:
snsTopicArnArg}));
    console.log("Successfully deleted.")
 }

 const displayBlockInfo = async (block) => {
   console.log(`Block ID: ${block.Id}`)
   console.log(`Block Type: ${block.BlockType}`)
   if (String(block).includes(String("EntityTypes"))){
       console.log(`EntityTypes: ${block.EntityTypes}`)
   }
   if (String(block).includes(String("Text"))){
       console.log(`EntityTypes: ${block.Text}`)
   }
   if (!String(block.BlockType).includes('PAGE')){
       console.log(`Confidence: ${block.Confidence}`)
   }
   console.log(`Page: ${block.Page}`)
   if (String(block.BlockType).includes("CELL")){
       console.log("Cell Information")
       console.log(`Column: ${block.ColumnIndex}`)
       console.log(`Row: ${block.RowIndex}`)
       console.log(`Column Span: ${block.ColumnSpan}`)
       console.log(`Row Span: ${block.RowSpan}`)
       if (String(block).includes("Relationships")){
           console.log(`Relationships: ${block.Relationships}`)
       }
   }

   console.log("Geometry")
   console.log(`Bounding Box: ${JSON.stringify(block.Geometry.BoundingBox)}`)
   console.log(`Polygon: ${JSON.stringify(block.Geometry.Polygon)}`)

   if (String(block.BlockType).includes('SELECTION_ELEMENT')){
     console.log('Selection Element detected:')
     if (String(block.SelectionStatus).includes('SELECTED')){
       console.log('Selected')
     } else {
       console.log('Not Selected')
     }

   }
```

```
 }

 const GetResults = async (processType, JobID) => {

   var maxResults = 1000
   var paginationToken = null
   var finished = false

   while (finished == false){
     var response = null
     if (processType == 'ANALYSIS'){
       if (paginationToken == null){
         response = textractClient.send(new
GetDocumentAnalysisCommand({JobId:JobID, MaxResults:maxResults}))

       }else{
         response = textractClient.send(new
GetDocumentAnalysisCommand({JobId:JobID, MaxResults:maxResults,
NextToken:paginationToken}))
       }
     }

     if(processType == 'DETECTION'){
       if (paginationToken == null){
         response = textractClient.send(new
GetDocumentTextDetectionCommand({JobId:JobID, MaxResults:maxResults}))

       }else{
         response = textractClient.send(new
GetDocumentTextDetectionCommand({JobId:JobID, MaxResults:maxResults,
NextToken:paginationToken}))
       }
     }

     await new Promise(resolve => setTimeout(resolve, 5000));
     console.log("Detected Documented Text")
     console.log(response)
     //console.log(Object.keys(response))
     console.log(typeof(response))
     var blocks = (await response).Blocks
     console.log(blocks)
     console.log(typeof(blocks))
     var docMetadata = (await response).DocumentMetadata
     var blockString = JSON.stringify(blocks)
```

```
      var parsed = JSON.parse(JSON.stringify(blocks))
      console.log(Object.keys(blocks))
      console.log(`Pages: ${docMetadata.Pages}`)
      blocks.forEach((block)=> {
        displayBlockInfo(block)
        console.log()
        console.log()
      })

      //console.log(blocks[0].BlockType)
      //console.log(blocks[1].BlockType)


      if(String(response).includes("NextToken")){
        paginationToken = response.NextToken
      }else{
        finished = true
      }
    }

  }


  // DELETE TOPIC AND QUEUE
  const main = async () => {
    var sqsAndTopic = await createTopicandQueue();
    var process = await processDocumment(processType, bucket, documentName,
 roleArn, sqsAndTopic[0], sqsAndTopic[1])
    var deleteResults = await deleteTopicAndQueue(sqsAndTopic[0],
 sqsAndTopic[1])
  }

main()
```

4. 執行程式碼。此操作可能需要一些時間來完成。完成之後，將顯示偵測到或分析的文字塊清單。

Amazon Textract 結果通知

Amazon Textract 將 Amazon Textract 分析請求結果發佈至 Amazon SNS Simple Notification Service (Amazon SNS) 主題，其中包括完成狀態。若要從 Amazon SNS 主題取得通知，請使用 Amazon SQS 佇列或AWS Lambda函數。如需詳細資訊，請參閱 調用 Amazon Textract 異步操作。如需範例，請參閱 檢測或分析多頁文檔中的文本。

結果的 JSON 格式如下：

```
{
  "JobId": "String",
  "Status": "String",
  "API": "String",
  "JobTag": "String",
  "Timestamp": Number,
  "DocumentLocation": {
    "S3ObjectName": "String",
    "S3Bucket": "String"
  }
}
```

此表格描述 Amazon Textract 應中的不同參數。

參數	描述
JobId	Amazon Textract 為作業分配的唯一編碼。它會與從`Start`操作，例如StartDocumentTextDetection。
狀態	任務的狀態。有效值包括「成功」、「失敗」或「錯誤」。
API	Amazon Textract 操作用於分析輸入文件，例如StartDocumentTextDetection或者StartDocumentAnalysis。
JobTag	用戶為任務指定的標識符。您指定`JobTag`在調用`Start`操作，例如StartDocumentTextDetection。
時間戳記	指示作業完成時間的 Unix 時間戳，以毫秒為單位返回。
文檔位置	已完成處理的文件的詳細資訊。包含檔案名稱與檔案所存放的 Amazon S3 儲存貯體。

處理受限制的呼叫和已刪除的連接

如果您超過每秒最大交易數 (TPS)，導致服務限制您的應用程序或連接斷開時，Amazon Textract 操作可能會失敗。例如，如果您在短時間內對 Amazon Textract 操作進行過多調用，則會限制您的呼叫並發送`ProvisionedThroughputExceededException`錯誤的操作響應。如需 Amazon Textract TPS 配額的詳細資訊，請參閱Amazon Textract 配額。

您可以通過自動重試操作來管理限制和斷開的連接。您可以指定重試次數，方法是在`Config`參數，當您創建 Amazon Textract 客户端時。我們建議重試計數 5。所以此AWSSDK 在失敗和拋出異常之前按指定次數重試操作。如需詳細資訊，請參閱 AWS 中的錯誤重試與指數退避。

> **Note**
>
> 自動重試適用於同步操作和異步操作。在指定自動重試之前，請確保您擁有最新版本的 AWS 開發工具包。如需詳細資訊，請參閱 步驟 2：設定AWS CLI和AWS開發套件。

以下範例說明如何在您處理多個檔案時自動重試 Amazon Textract 操作。

先決條件

- 如果您尚未：
  a. 使用`AmazonTextractFullAccess`和`AmazonS3ReadOnlyAccess`許可。如需詳細資訊，請參閱 步驟 1：設定 AWS 帳户並建立 IAM 使用者。
  b. 安裝並設定 AWS CLI 和 AWS SDK。如需詳細資訊，請參閱 步驟 2：設定AWS CLI和AWS開發套件。

自動重試操作

1. 將多個檔案映像上傳至 S3 儲存貯體以運行同步範例。將多頁面檔上傳至 S3 儲存貯體，然後運行`StartDocumentTextDetection`來運行異步示例。

   如需說明，請參閱「」將數據元上傳至 Amazon S3中的Amazon Simple Storage Service 用户指南。
2. 以下範例演示如何使用`Config`參數自動重試操作。同步示例調用`DetectDocumentText`操作，而異步示例調用`GetDocumentTextDetection`operation.

Sync Example

使用以下範例來調用DetectDocumentText操作，以取代您 Amazon S3 儲存貯體中的檔案。Inmain中，變更的值bucket至您的 S3 儲存貯體。變更的值documents設置為您在步驟 2 中上傳的文檔圖像的名稱。

```
import boto3
from botocore.client import Config
# Documents

def process_multiple_documents(bucket, documents):

    config = Config(retries = dict(max_attempts = 5))

    # Amazon Textract client
    textract = boto3.client('textract', config=config)

    for documentName in documents:

        print("\nProcessing:
 {}\n=========================================".format(documentName))

        # Call Amazon Textract
        response = textract.detect_document_text(
            Document={
                'S3Object': {
                    'Bucket': bucket,
                    'Name': documentName
                }
            })

        # Print detected text
        for item in response["Blocks"]:
            if item["BlockType"] == "LINE":
                print ('\033[94m' +  item["Text"] + '\033[0m')


def main():
    bucket = ""
    documents = ["document-image-1.png",
    "document-image-2.png", "document-image-3.png",
    "document-image-4.png", "document-image-5.png" ]
```

```
    process_multiple_documents(bucket, documents)



if __name__ == "__main__":
    main()
```

Async Example

使用下列範例來呼叫 GetDocumentTextDetection 操作。它假設您已經調用StartDocumentTextDetection取代為您 Amazon S3 儲存貯體中的檔案，並獲得JobId。Inmain中，變更的值bucket變更到您的 S3 儲存貯體中，並將roleArn添加到分配給文本角色的 Arn。您還需要變更的值document變更為您 Amazon S3 儲存貯體中的多頁面檔名稱。最後，取代region_name取代為您所在地區的名稱，並提供GetResults函數的名稱jobId。

```
import boto3
from botocore.client import Config

class DocumentProcessor:
    jobId = ''
    region_name = ''

    roleArn = ''
    bucket = ''
    document = ''

    sqsQueueUrl = ''
    snsTopicArn = ''
    processType = ''

    def __init__(self, role, bucket, document, region):
        self.roleArn = role
        self.bucket = bucket
        self.document = document
        self.region_name = region
        self.config = Config(retries = dict(max_attempts = 5))

        self.textract = boto3.client('textract', region_name=self.region_name,
 config=self.config)
        self.sqs = boto3.client('sqs')
        self.sns = boto3.client('sns')
```

```
# Display information about a block
    def DisplayBlockInfo(self, block):

        print("Block Id: " + block['Id'])
        print("Type: " + block['BlockType'])
        if 'EntityTypes' in block:
            print('EntityTypes: {}'.format(block['EntityTypes']))

        if 'Text' in block:
            print("Text: " + block['Text'])

        if block['BlockType'] != 'PAGE':
            print("Confidence: " + "{:.2f}".format(block['Confidence']) + "%")

        print('Page: {}'.format(block['Page']))

        if block['BlockType'] == 'CELL':
            print('Cell Information')
            print('\tColumn: {} '.format(block['ColumnIndex']))
            print('\tRow: {}'.format(block['RowIndex']))
            print('\tColumn span: {} '.format(block['ColumnSpan']))
            print('\tRow span: {}'.format(block['RowSpan']))

            if 'Relationships' in block:
                print('\tRelationships: {}'.format(block['Relationships']))

        print('Geometry')
        print('\tBounding Box: {}'.format(block['Geometry']['BoundingBox']))
        print('\tPolygon: {}'.format(block['Geometry']['Polygon']))

        if block['BlockType'] == 'SELECTION_ELEMENT':
            print('    Selection element detected: ', end='')
            if block['SelectionStatus'] == 'SELECTED':
                print('Selected')
            else:
                print('Not selected')

    def GetResults(self, jobId):
        maxResults = 1000
        paginationToken = None
        finished = False

        while finished == False:
```

```
            response = None

            if paginationToken == None:
                response =
 self.textract.get_document_text_detection(JobId=jobId,

 MaxResults=maxResults)
            else:
                response =
 self.textract.get_document_text_detection(JobId=jobId,

 MaxResults=maxResults,

 NextToken=paginationToken)

            blocks = response['Blocks']
            print('Detected Document Text')
            print('Pages: {}'.format(response['DocumentMetadata']['Pages']))

            # Display block information
            for block in blocks:
                self.DisplayBlockInfo(block)
                print()
                print()

            if 'NextToken' in response:
                paginationToken = response['NextToken']
            else:
                finished = True

def main():
    roleArn = 'role-arn'
    bucket = 'bucket-name'
    document = 'document-name'
    region_name = 'region-name'
    analyzer = DocumentProcessor(roleArn, bucket, document, region_name)
    analyzer.GetResults("job-id")

if __name__ == "__main__":
    main()
```

Amazon Textract 的最佳實務

Amazon Textract 使用機器學習來閱讀文檔，就像個人一樣。它從文檔中提取文本、表格和表單。使用下列最佳實務，以取得文檔的最佳結果。

提供最佳輸入文檔

以下是優化輸入文檔以獲得更好結果的幾種方法的列表。

- 確保您的文檔文本使用 Amazon Textract 支持的語言。目前，Amazon Textract 支援英文、西班牙文、德文、義大利文、法文和葡萄牙文。
- 提供高質量的圖像，理想情況下至少為 150 DPI。
- 如果您的文檔已採用 Amazon Textract 支持的文件格式之一（PDF、TIFF、JPEG 和 PNG），請不要在將文檔上傳到 Amazon Textract 之前對文檔進行轉換或縮小樣本。

為了在從文檔中的表中提取文本時獲得最佳結果，請確保：

- 文檔中的表格在視覺上與頁面上的周圍元素分離。例如，表格不會疊加到圖像或複雜圖案上。
- 表格中的文本是直立的。例如，文本不會相對於頁面上的其他文本進行旋轉。

從表中提取文本時，在以下情況下可能會看到不一致的結果：

- 跨多個列的合併表格單元格。
- 具有與同一表格的其他部分不同的單元格、行或列的表。

建議您使用文字偵測作為解決方法。

使用可信度分數

您應該考慮 Amazon Textract API 操作返回的置信度分數及其使用案例的敏感性。可信度分數是介於 0 到 100 之間的數字，表示給定預測正確性的概率。它可以幫助您就如何使用結果做出明智的決策。

在對檢測錯誤（誤報）敏感的應用程序中，強制執行最小置信度閾值。應用程序應該放棄低於該閾值的結果，或者將需要更高級別的人工審查的情況標記為情況。

最佳閾值取決於應用程式。對於存檔目的，例如記錄手寫筆記，它可能低至 50%。涉及財務決策的業務流程可能需要 90% 或更高的閾值。

考慮使用人工檢索

還可以考慮將人工審核納入工作流程。這對敏感應用程序尤其重要，例如涉及財務決策的業務流程。

教學課程

the section called “Block”從 Amazon Textract 操作返回的對象包含文本檢測和文本分析操作的結果，例如the section called “AnalyzeDocument”。以下 Python 教程顯示您可以使用塊對象的一些不同方式。例如，您可以將表格信息導出至以逗號分隔值 (CSV) 檔案。

這些教程使用返回所有結果的同步 Amazon Textract 操作。如果你想使用異步操作，如the section called “StartDocumentAnalysis”，您需要更改示例代碼以適應多個返回的`Block`物件。要使用異步操作示例，請確保您已按照為異步操作配置 Amazon Textract。

有關向您展示使用 Amazon Textract 的其他方式的示例，請參閱其他程式碼範例。

主題

- 先決條件
- 從表單文檔中提取鍵值對
- 將資料表導出為 CSV 檔
- 建立AWS Lambda函式
- 其他程式碼範例

先決條件

您必須先配置您的環境，然後才能運行本節中的範例。

若要設定您的環境

1. 使用建立或更新 IAM 用户`AmazonTextractFullAccess`許可。如需詳細資訊，請參閱 步驟 1：設定 AWS 帳户並建立 IAM 使用者。
2. 安裝並設定 AWS CLI 和 AWS SDK。如需詳細資訊，請參閱 步驟 2：設定AWS CLI和AWS開發套件。

從表單文檔中提取鍵值對

以下 Python 範例示範如何從the section called “Block”儲存在地圖中的物件。塊對象從調用the section called “AnalyzeDocument”。如需詳細資訊，請參閱 表單數據（鍵值對）。

您可以使用以下函數：

- get_kv_map— 呼叫AnalyzeDocument，並將鍵和值塊對象存儲在地圖中。
- get_kv_relationship和find_value_block— 從地圖構造鍵值關係。

從表單檔中提取鍵-值對

1. 設定您的環境。如需詳細資訊，請參閱 先決條件。
2. 將下列範本程式碼複製到名為的textract_python_kv_parser.py。

```
import boto3
import sys
import re
import json


def get_kv_map(file_name):

    with open(file_name, 'rb') as file:
        img_test = file.read()
        bytes_test = bytearray(img_test)
        print('Image loaded', file_name)

    # process using image bytes
    client = boto3.client('textract')
    response = client.analyze_document(Document={'Bytes': bytes_test},
 FeatureTypes=['FORMS'])

    # Get the text blocks
    blocks=response['Blocks']


    # get key and value maps
    key_map = {}
    value_map = {}
    block_map = {}
    for block in blocks:
        block_id = block['Id']
        block_map[block_id] = block
        if block['BlockType'] == "KEY_VALUE_SET":
            if 'KEY' in block['EntityTypes']:
                key_map[block_id] = block
            else:
                value_map[block_id] = block
```

```
    return key_map, value_map, block_map


def get_kv_relationship(key_map, value_map, block_map):
    kvs = {}
    for block_id, key_block in key_map.items():
        value_block = find_value_block(key_block, value_map)
        key = get_text(key_block, block_map)
        val = get_text(value_block, block_map)
        kvs[key] = val
    return kvs


def find_value_block(key_block, value_map):
    for relationship in key_block['Relationships']:
        if relationship['Type'] == 'VALUE':
            for value_id in relationship['Ids']:
                value_block = value_map[value_id]
    return value_block


def get_text(result, blocks_map):
    text = ''
    if 'Relationships' in result:
        for relationship in result['Relationships']:
            if relationship['Type'] == 'CHILD':
                for child_id in relationship['Ids']:
                    word = blocks_map[child_id]
                    if word['BlockType'] == 'WORD':
                        text += word['Text'] + ' '
                    if word['BlockType'] == 'SELECTION_ELEMENT':
                        if word['SelectionStatus'] == 'SELECTED':
                            text += 'X '


    return text


def print_kvs(kvs):
    for key, value in kvs.items():
        print(key, ":", value)
```

```
def search_value(kvs, search_key):
    for key, value in kvs.items():
        if re.search(search_key, key, re.IGNORECASE):
            return value

def main(file_name):

    key_map, value_map, block_map = get_kv_map(file_name)

    # Get Key Value relationship
    kvs = get_kv_relationship(key_map, value_map, block_map)
    print("\n\n== FOUND KEY : VALUE pairs ===\n")
    print_kvs(kvs)

    # Start searching a key value
    while input('\n Do you want to search a value for a key? (enter "n" for exit)
 ') != 'n':
        search_key = input('\n Enter a search key:')
        print('The value is:', search_value(kvs, search_key))

if __name__ == "__main__":
    file_name = sys.argv[1]
    main(file_name)
```

3. 在命令提示中，輸入以下命令：Replacefile取代為您要分析的檔案圖像檔案。

```
textract_python_kv_parser.py file
```

4. 當系統提示時，輸入輸入文檔中的鍵。如果代碼檢測到密鑰，它將顯示密鑰的值。

將資料表導出為 CSV 檔

這些 Python 範例示範如何將表格從文件圖像導出至以逗號分隔值 (CSV) 檔案。

同步文檔分析示例從調用the section called “AnalyzeDocument”。異步文檔分析示例調用the section called “StartDocumentAnalysis”，然後從the section called “GetDocumentAnalysis”如Block物件。

表信息返回為the section called “Block”對象從調用the section called “AnalyzeDocument”。如需詳細資訊，請參閱 資料表。所以此Block對象存儲在用於將表數據導出到 CSV 文件的地圖結構中。

Synchronous

在此示例中，您將使用以下函數：

- get_table_csv_results— 呼叫AnalyzeDocument，並構建在文檔中檢測到的表的映射。創建所有檢測到的表的 CSV 表示形式。
- generate_table_csv— 為單個表生成 CSV 文件。
- get_rows_columns_map— 從地圖中獲取行和列。
- get_text— 從單元格獲取文本。

將表導出到 CSV 文件

1. 設定您的環境。如需詳細資訊，請參閱 先決條件。
2. 將下列範本程式碼複製到名為的textract_python_table_parser.py。

```
import webbrowser, os
import json
import boto3
import io
from io import BytesIO
import sys
from pprint import pprint


def get_rows_columns_map(table_result, blocks_map):
    rows = {}
    for relationship in table_result['Relationships']:
        if relationship['Type'] == 'CHILD':
            for child_id in relationship['Ids']:
                cell = blocks_map[child_id]
                if cell['BlockType'] == 'CELL':
                    row_index = cell['RowIndex']
                    col_index = cell['ColumnIndex']
                    if row_index not in rows:
                        # create new row
                        rows[row_index] = {}

                    # get the text value
                    rows[row_index][col_index] = get_text(cell, blocks_map)
    return rows
```

```
def get_text(result, blocks_map):
    text = ''
    if 'Relationships' in result:
        for relationship in result['Relationships']:
            if relationship['Type'] == 'CHILD':
                for child_id in relationship['Ids']:
                    word = blocks_map[child_id]
                    if word['BlockType'] == 'WORD':
                        text += word['Text'] + ' '
                    if word['BlockType'] == 'SELECTION_ELEMENT':
                        if word['SelectionStatus'] =='SELECTED':
                            text +=  'X '
    return text


def get_table_csv_results(file_name):

    with open(file_name, 'rb') as file:
        img_test = file.read()
        bytes_test = bytearray(img_test)
        print('Image loaded', file_name)

    # process using image bytes
    # get the results
    client = boto3.client('textract')

    response = client.analyze_document(Document={'Bytes': bytes_test},
 FeatureTypes=['TABLES'])

    # Get the text blocks
    blocks=response['Blocks']
    pprint(blocks)

    blocks_map = {}
    table_blocks = []
    for block in blocks:
        blocks_map[block['Id']] = block
        if block['BlockType'] == "TABLE":
            table_blocks.append(block)

    if len(table_blocks) <= 0:
        return "<b> NO Table FOUND </b>"
```

```
    csv = ''
    for index, table in enumerate(table_blocks):
        csv += generate_table_csv(table, blocks_map, index +1)
        csv += '\n\n'

    return csv

def generate_table_csv(table_result, blocks_map, table_index):
    rows = get_rows_columns_map(table_result, blocks_map)

    table_id = 'Table_' + str(table_index)

    # get cells.
    csv = 'Table: {0}\n\n'.format(table_id)

    for row_index, cols in rows.items():

        for col_index, text in cols.items():
            csv += '{}'.format(text) + ","
        csv += '\n'

    csv += '\n\n\n'
    return csv

def main(file_name):
    table_csv = get_table_csv_results(file_name)

    output_file = 'output.csv'

    # replace content
    with open(output_file, "wt") as fout:
        fout.write(table_csv)

    # show the results
    print('CSV OUTPUT FILE: ', output_file)


if __name__ == "__main__":
    file_name = sys.argv[1]
    main(file_name)
```

3. 在命令提示中，輸入以下命令：Replacefile以您要分析的檔案圖像文件名稱取代。

```
python textract_python_table_parser.py file
```

運行該示例時，CSV 輸出將保存在名為output.csv。

Asynchronous

在本示例中，您將使用兩個不同的腳本。第一個腳本開始異步分析文檔的過程StartDocumentAnalysis並獲取Block返回的信息GetDocumentAnalysis。第二個腳本採用返回的Block信息，將數據格式化為表格，並將表保存到 CSV 文件中。

將表導出到 CSV 文件

1. 設定您的環境。如需詳細資訊，請參閱 先決條件。
2. 確保您已遵循為異步操作配置 Amazon Textract。通過該頁上記錄的過程，您可以發送和接收有關異步作業完成狀態的消息。
3. 在下面的代碼示例中，替換roleArn並將 Arn 分配給您在步驟 2 中建立的角色。替換bucket以包含您的檔案的 S3 儲存體名稱取代。替換document取代為 S3 儲存貯體中的檔案名稱。替換region_name取代為儲存貯體的區域名稱。

   將下列範本程式碼複製到名為的start_doc_analysis_for_table_extraction.py。。

```
import boto3
import time

class DocumentProcessor:

    jobId = ''
    region_name = ''

    roleArn = ''
    bucket = ''
    document = ''

    sqsQueueUrl = ''
    snsTopicArn = ''
    processType = ''

    def __init__(self, role, bucket, document, region):
        self.roleArn = role
        self.bucket = bucket
```

```
        self.document = document
        self.region_name = region

        self.textract = boto3.client('textract', region_name=self.region_name)
        self.sqs = boto3.client('sqs')
        self.sns = boto3.client('sns')

    def ProcessDocument(self):

        jobFound = False

        response =
 self.textract.start_document_analysis(DocumentLocation={'S3Object': {'Bucket':
 self.bucket, 'Name': self.document}},
                FeatureTypes=["TABLES", "FORMS"],
 NotificationChannel={'RoleArn': self.roleArn, 'SNSTopicArn':
 self.snsTopicArn})
        print('Processing type: Analysis')

        print('Start Job Id: ' + response['JobId'])

        print('Done!')

    def CreateTopicandQueue(self):

        millis = str(int(round(time.time() * 1000)))

        # Create SNS topic
        snsTopicName = "AmazonTextractTopic" + millis

        topicResponse = self.sns.create_topic(Name=snsTopicName)
        self.snsTopicArn = topicResponse['TopicArn']

        # create SQS queue
        sqsQueueName = "AmazonTextractQueue" + millis
        self.sqs.create_queue(QueueName=sqsQueueName)
        self.sqsQueueUrl = self.sqs.get_queue_url(QueueName=sqsQueueName)
['QueueUrl']

        attribs = self.sqs.get_queue_attributes(QueueUrl=self.sqsQueueUrl,
                                                AttributeNames=['QueueArn'])
['Attributes']

        sqsQueueArn = attribs['QueueArn']
```

```
        # Subscribe SQS queue to SNS topic
        self.sns.subscribe(TopicArn=self.snsTopicArn, Protocol='sqs',
 Endpoint=sqsQueueArn)

        # Authorize SNS to write SQS queue
        policy = """{{
      "Version":"2012-10-17",
      "Statement":[
        {{
          "Sid":"MyPolicy",
          "Effect":"Allow",
          "Principal" : {{"AWS" : "*"}},
          "Action":"SQS:SendMessage",
          "Resource": "{}",
          "Condition":{{
            "ArnEquals":{{
              "aws:SourceArn": "{}"
            }}
          }}
        }}
      ]
    }}""".format(sqsQueueArn, self.snsTopicArn)

        response = self.sqs.set_queue_attributes(
            QueueUrl=self.sqsQueueUrl,
            Attributes={
                'Policy': policy
            })

def main():
    roleArn = 'role-arn'
    bucket = 'bucket-name'
    document = 'document-name'
    region_name = 'region-name'

    analyzer = DocumentProcessor(roleArn, bucket, document, region_name)
    analyzer.CreateTopicandQueue()
    analyzer.ProcessDocument()

if __name__ == "__main__":
    main()
```

4. 執行程式碼。代碼將打印一個 JobId。將此 JobId 複製下來。

5. 等待您的作業完成處理，並在完成後，將以下代碼複製到名為get_doc_analysis_for_table_extraction.py。替換`jobId`使用您之前複製的 Job ID。替換`region_name`，其中包含與您的 Textract 角色關聯的區域的名稱。替換`file_name`取代為您想要指定輸出 CSV 的名稱。

```
import boto3
from pprint import pprint

jobId = 'job-id'
region_name = 'region-name'
file_name = "output-file-name.csv"

textract = boto3.client('textract', region_name=region_name)

# Display information about a block
def DisplayBlockInfo(block):
    print("Block Id: " + block['Id'])
    print("Type: " + block['BlockType'])
    if 'EntityTypes' in block:
        print('EntityTypes: {}'.format(block['EntityTypes']))

    if 'Text' in block:
        print("Text: " + block['Text'])

    if block['BlockType'] != 'PAGE':
        print("Confidence: " + "{:.2f}".format(block['Confidence']) + "%")

def GetResults(jobId, file_name):
    maxResults = 1000
    paginationToken = None
    finished = False

    while finished == False:

        response = None

        if paginationToken == None:
            response = textract.get_document_analysis(JobId=jobId,
 MaxResults=maxResults)
        else:
            response = textract.get_document_analysis(JobId=jobId,
 MaxResults=maxResults,
```

```
 NextToken=paginationToken)

        blocks = response['Blocks']
        table_csv = get_table_csv_results(blocks)
        output_file = file_name
        # replace content
        with open(output_file, "at") as fout:
            fout.write(table_csv)
        # show the results
        print('Detected Document Text')
        print('Pages: {}'.format(response['DocumentMetadata']['Pages']))
        print('OUTPUT TO CSV FILE: ', output_file)

        # Display block information
        for block in blocks:
            DisplayBlockInfo(block)
            print()
            print()

        if 'NextToken' in response:
            paginationToken = response['NextToken']
        else:
            finished = True


def get_rows_columns_map(table_result, blocks_map):
    rows = {}
    for relationship in table_result['Relationships']:
        if relationship['Type'] == 'CHILD':
            for child_id in relationship['Ids']:
                try:
                    cell = blocks_map[child_id]
                    if cell['BlockType'] == 'CELL':
                        row_index = cell['RowIndex']
                        col_index = cell['ColumnIndex']
                        if row_index not in rows:
                            # create new row
                            rows[row_index] = {}

                        # get the text value
                        rows[row_index][col_index] = get_text(cell, blocks_map)
                except KeyError:
                    print("Error extracting Table data - {}:".format(KeyError))
```

```
                    pass
    return rows


def get_text(result, blocks_map):
    text = ''
    if 'Relationships' in result:
        for relationship in result['Relationships']:
            if relationship['Type'] == 'CHILD':
                for child_id in relationship['Ids']:
                    try:
                        word = blocks_map[child_id]
                        if word['BlockType'] == 'WORD':
                            text += word['Text'] + ' '
                        if word['BlockType'] == 'SELECTION_ELEMENT':
                            if word['SelectionStatus'] == 'SELECTED':
                                text += 'X '
                    except KeyError:
                        print("Error extracting Table data -
 {}:".format(KeyError))

    return text


def get_table_csv_results(blocks):

    pprint(blocks)

    blocks_map = {}
    table_blocks = []
    for block in blocks:
        blocks_map[block['Id']] = block
        if block['BlockType'] == "TABLE":
            table_blocks.append(block)

    if len(table_blocks) <= 0:
        return "<b> NO Table FOUND </b>"

    csv = ''
    for index, table in enumerate(table_blocks):
        csv += generate_table_csv(table, blocks_map, index + 1)
        csv += '\n\n'

    return csv
```

```
def generate_table_csv(table_result, blocks_map, table_index):
    rows = get_rows_columns_map(table_result, blocks_map)

    table_id = 'Table_' + str(table_index)

    # get cells.
    csv = 'Table: {0}\n\n'.format(table_id)

    for row_index, cols in rows.items():

        for col_index, text in cols.items():
            csv += '{}'.format(text) + ","
        csv += '\n'

    csv += '\n\n\n'
    return csv

response_blocks = GetResults(jobId, file_name)
```

6. 執行程式碼。

   獲得結果後，請務必刪除相關的 SNS 和 SQS 資源，否則您可能會為這些資源產生費用。

建立AWS Lambda函式

您可以從AWS Lambda函數。以下說明顯示如何在 Python 中建立 Lambda 函數，以調用the section called “DetectDocumentText”。它會返回the section called “Block”物件。要運行此示例，您需要一個包含 PNG 或 JPEG 格式文檔的 Amazon S3 存儲桶。若要建立函數，您可以使用主控台。

有關使用 Lambda 函數大規模處理文檔的示例，請參閱使用 Amazon Textract 進行大規模文檔處理。

要從 Lambda 函數調用 DetectDocumentText 操作，請執行以下操作：

步驟 1：建立 Lambda 部署套件

1. 開啟命令視窗。
2. 輸入以下命令以最新版的AWS開發套件。

```
pip install boto3 --target python/.
```

```
zip boto3-layer.zip -r python/
```

步驟 2：建立 Lambda 函數

1. 請登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/lambda/ 的 AWS Lambda 主控台。
2. 選擇 Create function (建立函數)。
3. 指定下列內容。
   - 選擇 Author from scratch (從頭開始撰寫)。
   - 針對 Function name (函數名稱)，輸入名稱。
   - 適用於執行時間，選擇Python 3.7或者Python 3.6。
   - 適用於選擇或建立執行角色，選擇建立具備基本 Lambda 許可的新角色。
4. 選擇建立函數建立 Lambda 函數。
5. 在以下網址開啟 IAM 主控台：https://console.aws.amazon.com/iam/。
6. 在導覽窗格中，選取角色。
7. 從資源列表中選擇 Lambda 為您建立的 IAM 角色。角色名稱以 Lambda 函數名稱開頭。
8. 選擇Permissions (許可)選項卡，然後選擇連接政策。
9. 選擇亞 AmazonTextractFullAccess 和 AmazonS3ReadOnlyAccess 政策。
10. 選擇連接政策。

如需詳細資訊，請參閱「」使用主控台建立 Lambda 函數

步驟 3：建立和添加圖層

1. 於 https://console.aws.amazon.com/lambda/ 開啟 AWS Lambda 主控台。
2. 在導覽窗格中，選擇 Layers (層)。
3. 選擇 Create layer (建立 Layer)。
4. 適用於名稱中，輸入名稱。
5. 在 Description (描述) 中，輸入描述。
6. 適用於程式碼輸入類型，選擇上傳 .zip 檔案，然後選取上傳。
7. 在對話框中，選擇 zip 文件 (boto3-layer.zip)，即您在步驟 1：建立 Lambda 部署套件。

8. 適用於相容的運行時中，選擇您在步驟 2：建立 Lambda 函數。
9. 選擇建立建立圖層。
10. 選擇導航窗格菜單圖標。
11. 在導覽窗格中，選擇函數。
12. 在資源列表中，選擇您在步驟 2：建立 Lambda 函數。
13. 選擇組態並在設計師部分中，選擇圖層（在你的 Lambda 函數名稱下）。
14. 在 中圖層部分中，選擇新增圖層。
15. 選擇從運行時兼容圖層列表中進行選擇。
16. In相容的層中，選取名稱和版本在步驟 3 中建立的圖層。
17. 選擇 Add (新增)。

步驟 4：將 python 程式碼複製到函數

1. In設計師，選取函數。
2. 在函式程式碼編輯器中，將以下內容添加到lambda_function.py。更改bucket和document添加到您的存儲桶和文檔。

```
import json
import boto3

def lambda_handler(event, context):

    bucket="bucket"
    document="document"
    client = boto3.client('textract')



    #process using S3 object
    response = client.detect_document_text(
        Document={'S3Object': {'Bucket': bucket, 'Name': document}})

    #Get the text blocks
    blocks=response['Blocks']

    return {
        'statusCode': 200,
        'body': json.dumps(blocks)
```

```
    }
```

3. 選擇Save以保存您的 Lambda 函數。

步驟 5：測試您的 Lambda

1. 選擇測試。
2. 輸入Event name。
3. 選擇 Create (建立)。
4. 輸出，列表the section called “Block”對象將顯示在「執行結果」窗格中。

如果AWS Lambda函數返回超時錯誤，則可能導致 Amazon Textract API 操作調用。有關延長超時期限的信息AWS Lambda函數，請參閱AWS Lambda 功能。

有關從您的代碼調用 Lambda 函數的信息，請參閱呼叫AWS Lambda函數。

其他程式碼範例

下表提供連結到更多 Amazon Textract 碼範例。

範例	描述
Amazon Textract 碼範例	示範您可以使用 Amazon Textract 的各種方式。
使用 Amazon Textract 進行大規模文檔處理	顯示大規模處理文檔的無服務器參考體繫結構。
Amazon Textract 解析器	示範如何解析the section called “Block”Amazon Textract 操作返回的對象。
Amazon Textract 文檔代碼示例	本指南中使用的程式碼範例。
文字工具	演示如何將 Amazon Textract 輸出轉換為多種格式。
使用 Amazon Textract 生成可搜索的 PDF 文檔	演示如何從不同類型的輸入文檔（如 JPG/PNG 格式圖像和掃描的 PDF 文檔）創建可搜索的 PDF 文檔。

Amazon Textract 的程式碼範例

下列程式碼範例示範如何使用 Amazon Textract 搭配AWS軟件開發工具包 (SDK)。

這些範例分為下列類別：

動作

向您展示如何呼叫個別服務函數的程式碼摘錄。

跨服務範例

跨多個工作的示例應用程序AWS服務。

如需完整的清單AWSSDK 開發人員指南和代碼示例，請參閱將 Amazon Textract 與AWSSDK。本主題還包括有關入門信息和以前 SDK 版本的詳細信息。

程式碼範例

- Amazon Textract 的操作
  - 使用 Amazon Textract 和AWSSDK
  - 使用 Amazon Textract 和AWSSDK
  - 獲取有關 Amazon Textract 文檔分析作業的數據，使用AWSSDK
  - 使用亞馬遜 Textract 和AWSSDK
  - 啟動異步文本檢測使用 Amazon Textract 和AWSSDK
- Amazon Textract 的跨服務範例
  - 建立 Amazon Textract Explorer 應用程式
  - 檢測從圖像中提取的文本中使用AWSSDK

Amazon Textract 的操作

下列程式碼範例示範如 Amazon Textract 使用AWS開發套件。這些摘錄稱為 Amazon Textract API，不打算單獨運行。每個範例均包含 GitHub 的連結，您可以在連結中找到如何在內容中設定和運行程式碼的相關指示。

下列範例僅包含最常使用的動作。如需完整清單，請參Amazon Textract API 參考。

範例

- 使用 Amazon Textract 和AWSSDK
- 使用 Amazon Textract 和AWSSDK
- 獲取有關 Amazon Textract 文檔分析作業的數據，使用AWSSDK
- 使用亞馬遜 Textract 和AWSSDK
- 啟動異步文本檢測使用 Amazon Textract 和AWSSDK

使用 Amazon Textract 和AWSSDK

下列程式碼範例示範如何使用 Amazon Textract 分析文件。

Java

適用於 Java 2.x 的 SDK

```
   public static void analyzeDoc(TextractClient textractClient, String
sourceDoc) {

       try {
           InputStream sourceStream = new FileInputStream(new File(sourceDoc));
           SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

           // Get the input Document object as bytes
           Document myDoc = Document.builder()
                   .bytes(sourceBytes)
                   .build();

           List<FeatureType> featureTypes = new ArrayList<FeatureType>();
           featureTypes.add(FeatureType.FORMS);
           featureTypes.add(FeatureType.TABLES);

           AnalyzeDocumentRequest analyzeDocumentRequest =
AnalyzeDocumentRequest.builder()
                   .featureTypes(featureTypes)
                   .document(myDoc)
                   .build();

           AnalyzeDocumentResponse analyzeDocument =
textractClient.analyzeDocument(analyzeDocumentRequest);
           List<Block> docInfo = analyzeDocument.blocks();
```

```
            Iterator<Block> blockIterator = docInfo.iterator();

            while(blockIterator.hasNext()) {
                Block block = blockIterator.next();
                System.out.println("The block type is "
 +block.blockType().toString());
            }

        } catch (TextractException | FileNotFoundException e) {

            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- 在 GitHub 上尋找指示和更多程式碼。
- 如需 API 詳細資訊，請參AnalyzeDocument在AWS SDK for Java 2.xAPI 參考。

Python

適用於 Python (Boto3) 的 SDK

```
class TextractWrapper:
    """Encapsulates Textract functions."""
    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource

    def analyze_file(
            self, feature_types, *, document_file_name=None,
 document_bytes=None):
        """
        Detects text and additional elements, such as forms or tables, in a local
 image
        file or from in-memory byte data.
```

```
        The image must be in PNG or JPG format.

        :param feature_types: The types of additional document features to
detect.
        :param document_file_name: The name of a document image file.
        :param document_bytes: In-memory byte data of a document image.
        :return: The response from Amazon Textract, including a list of blocks
                 that describe elements detected in the image.
        """
        if document_file_name is not None:
            with open(document_file_name, 'rb') as document_file:
                document_bytes = document_file.read()
        try:
            response = self.textract_client.analyze_document(
                Document={'Bytes': document_bytes}, FeatureTypes=feature_types)
            logger.info(
                "Detected %s blocks.", len(response['Blocks']))
        except ClientError:
            logger.exception("Couldn't detect text.")
            raise
        else:
            return response
```

- 在 GitHub，尋找指示和更多代碼。
- 如需 API 詳細資訊，請參AnalyzeDocument在AWSSDK for Python (Boto3) API 參考。

如需完整的清單AWSSDK 開發人員指南和代碼示例，請參閱將 Amazon Textract 與AWSSDK。本主題還包括有關入門信息和以前 SDK 版本的詳細信息。

使用 Amazon Textract 和AWSSDK

下列程式碼範例示範如何使用 Amazon Textract 檢測文件中的文字。

Java

適用於 Java 2.x 的 SDK

檢測輸入文檔中的文本。

```
   public static void detectDocText(TextractClient textractClient,String
sourceDoc) {
```

```
        try {

            InputStream sourceStream = new FileInputStream(new File(sourceDoc));
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

            // Get the input Document object as bytes
            Document myDoc = Document.builder()
                    .bytes(sourceBytes)
                    .build();

            DetectDocumentTextRequest detectDocumentTextRequest =
DetectDocumentTextRequest.builder()
                    .document(myDoc)
                    .build();

            // Invoke the Detect operation
            DetectDocumentTextResponse textResponse =
textractClient.detectDocumentText(detectDocumentTextRequest);

            List<Block> docInfo = textResponse.blocks();

            Iterator<Block> blockIterator = docInfo.iterator();

            while(blockIterator.hasNext()) {
                Block block = blockIterator.next();
                System.out.println("The block type is "
+block.blockType().toString());
            }

            DocumentMetadata documentMetadata = textResponse.documentMetadata();
            System.out.println("The number of pages in the document is "
+documentMetadata.pages());

        } catch (TextractException | FileNotFoundException e) {

            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

檢測位於 Amazon S3 儲存貯體中的文件的文字。

```
   public static void detectDocTextS3 (TextractClient textractClient, String
bucketName, String docName) {

       try {
           S3Object s3Object = S3Object.builder()
                   .bucket(bucketName)
                   .name(docName)
                   .build();

           // Create a Document object and reference the s3Object instance
           Document myDoc = Document.builder()
                   .s3Object(s3Object)
                   .build();

           // Create a DetectDocumentTextRequest object
           DetectDocumentTextRequest detectDocumentTextRequest =
DetectDocumentTextRequest.builder()
                   .document(myDoc)
                   .build();

           // Invoke the detectDocumentText method
           DetectDocumentTextResponse textResponse =
textractClient.detectDocumentText(detectDocumentTextRequest);

           List<Block> docInfo = textResponse.blocks();

           Iterator<Block> blockIterator = docInfo.iterator();

           while(blockIterator.hasNext()) {
               Block block = blockIterator.next();
               System.out.println("The block type is "
+block.blockType().toString());
           }

           DocumentMetadata documentMetadata = textResponse.documentMetadata();
           System.out.println("The number of pages in the document is "
+documentMetadata.pages());

       } catch (TextractException e) {

           System.err.println(e.getMessage());
           System.exit(1);
       }
```

```
    }
```

- 在 GitHub 上尋找指示和更多程式碼。
- 如需 API 詳細資訊，請參DetectDocumentText在AWS SDK for Java 2.xAPI 參考。

Python

適用於 Python (Boto3) 的 SDK

```
class TextractWrapper:
    """Encapsulates Textract functions."""
    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource

    def detect_file_text(self, *, document_file_name=None, document_bytes=None):
        """
        Detects text elements in a local image file or from in-memory byte data.
        The image must be in PNG or JPG format.

        :param document_file_name: The name of a document image file.
        :param document_bytes: In-memory byte data of a document image.
        :return: The response from Amazon Textract, including a list of blocks
                 that describe elements detected in the image.
        """
        if document_file_name is not None:
            with open(document_file_name, 'rb') as document_file:
                document_bytes = document_file.read()
        try:
            response = self.textract_client.detect_document_text(
                Document={'Bytes': document_bytes})
            logger.info(
                "Detected %s blocks.", len(response['Blocks']))
        except ClientError:
            logger.exception("Couldn't detect text.")
```

```
            raise
        else:
            return response
```

- 在 GitHub，尋找指示和更多代碼。
- 如需 API 詳細資訊，請參DetectDocumentText在AWSSDK for Python (Boto3) API 參考。

如需完整的清單AWSSDK 開發人員指南和代碼示例，請參閱將 Amazon Textract 與AWSSDK。本主題還包括有關入門信息和以前 SDK 版本的詳細信息。

獲取有關 Amazon Textract 文檔分析作業的數據，使用AWSSDK

下列程式碼範例示範如何獲取 Amazon Textract 文件分析任務的相關資訊。

Python

適用於 Python (Boto3) 的 SDK

```
class TextractWrapper:
    """Encapsulates Textract functions."""
    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource

    def get_analysis_job(self, job_id):
        """
        Gets data for a previously started detection job that includes additional
        elements.

        :param job_id: The ID of the job to retrieve.
        :return: The job data, including a list of blocks that describe elements
                 detected in the image.
        """
        try:
```

```
            response = self.textract_client.get_document_analysis(
                JobId=job_id)
            job_status = response['JobStatus']
            logger.info("Job %s status is %s.", job_id, job_status)
        except ClientError:
            logger.exception("Couldn't get data for job %s.", job_id)
            raise
        else:
            return response
```

- 在 GitHub，尋找指示和更多代碼。
- 如需 API 詳細資訊，請參GetDocumentAnalysis在AWSSDK for Python (Boto3) API 參考。

如需完整的清單AWSSDK 開發人員指南和代碼示例，請參閱將 Amazon Textract 與AWSSDK。本主題還包括有關入門信息和以前 SDK 版本的詳細信息。

使用亞馬遜 Textract 和AWSSDK

下列程式碼範例示範如何使用 Amazon Textract 開始非同步分析文件。

Java

適用於 Java 2.x 的 SDK

```
   public static String startDocAnalysisS3 (TextractClient textractClient,
String bucketName, String docName) {

       try {

           List<FeatureType> myList = new ArrayList<FeatureType>();
           myList.add(FeatureType.TABLES);
           myList.add(FeatureType.FORMS);

           S3Object s3Object = S3Object.builder()
                   .bucket(bucketName)
                   .name(docName)
                   .build();

           DocumentLocation location = DocumentLocation.builder()
                   .s3Object(s3Object)
```

```
                    .build();

            StartDocumentAnalysisRequest documentAnalysisRequest =
StartDocumentAnalysisRequest.builder()
                    .documentLocation(location)
                    .featureTypes(myList)
                    .build();

            StartDocumentAnalysisResponse response =
textractClient.startDocumentAnalysis(documentAnalysisRequest);

            // Get the job ID
            String jobId = response.jobId();
            return jobId;

        } catch (TextractException e) {

            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "" ;
    }

    private static String getJobResults(TextractClient textractClient, String
jobId) {

        boolean finished = false;
        int index = 0 ;
        String status = "" ;

       try {
        while (!finished) {
            GetDocumentAnalysisRequest analysisRequest =
GetDocumentAnalysisRequest.builder()
                    .jobId(jobId)
                    .maxResults(1000)
                    .build();

            GetDocumentAnalysisResponse response =
textractClient.getDocumentAnalysis(analysisRequest);
            status = response.jobStatus().toString();

            if (status.compareTo("SUCCEEDED") == 0)
                finished = true;
```

```
            else {
                System.out.println(index + " status is: " + status);
                Thread.sleep(1000);
            }
            index++ ;
        }
        return status;

       } catch( InterruptedException e) {
           System.out.println(e.getMessage());
           System.exit(1);
       }
       return "";
    }
```

- 在 GitHub 上尋找指示和更多程式碼。
- 如需 API 詳細資訊，請參StartDocumentAnalysis在AWS SDK for Java 2.xAPI 參考。

Python

適用於 Python (Boto3) 的 SDK

啟動異步作業以分析文檔。

```
class TextractWrapper:
    """Encapsulates Textract functions."""
    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource

    def start_analysis_job(
            self, bucket_name, document_file_name, feature_types, sns_topic_arn,
            sns_role_arn):
        """
```

```
        Starts an asynchronous job to detect text and additional elements, such
as
        forms or tables, in an image stored in an Amazon S3 bucket. Textract
publishes
        a notification to the specified Amazon SNS topic when the job completes.
        The image must be in PNG, JPG, or PDF format.

        :param bucket_name: The name of the Amazon S3 bucket that contains the
image.
        :param document_file_name: The name of the document image stored in
Amazon S3.
        :param feature_types: The types of additional document features to
detect.
        :param sns_topic_arn: The Amazon Resource Name (ARN) of an Amazon SNS
topic
                              where job completion notification is published.
        :param sns_role_arn: The ARN of an AWS Identity and Access Management
(IAM)
                             role that can be assumed by Textract and grants
permission
                             to publish to the Amazon SNS topic.
        :return: The ID of the job.
        """
        try:
            response = self.textract_client.start_document_analysis(
                DocumentLocation={
                    'S3Object': {'Bucket': bucket_name, 'Name':
document_file_name}},
                NotificationChannel={
                    'SNSTopicArn': sns_topic_arn, 'RoleArn': sns_role_arn},
                FeatureTypes=feature_types)
            job_id = response['JobId']
            logger.info(
                "Started text analysis job %s on %s.", job_id,
document_file_name)
        except ClientError:
            logger.exception("Couldn't analyze text in %s.", document_file_name)
            raise
        else:
            return job_id
```

- 在 GitHub 上尋找指示和更多程式碼。

- 如需 API 詳細資訊，請參StartDocumentAnalysis在AWSSDK for Python (Boto3) API 參考。

如需完整的清單AWSSDK 開發人員指南和代碼示例，請參閱將 Amazon Textract 與AWSSDK。本主題還包括有關入門信息和以前 SDK 版本的詳細信息。

啟動異步文本檢測使用 Amazon Textract 和AWSSDK

下列程式碼範例示範如何使用 Amazon Textract 在文件中啟動非同步文字檢測。

Python

適用於 Python (Boto3) 的 SDK

啟動非同步任務，以檢測文件中的文字。

```
class TextractWrapper:
    """Encapsulates Textract functions."""
    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource

    def start_detection_job(
            self, bucket_name, document_file_name, sns_topic_arn, sns_role_arn):
        """
        Starts an asynchronous job to detect text elements in an image stored in
 an
        Amazon S3 bucket. Textract publishes a notification to the specified
 Amazon SNS
        topic when the job completes.
        The image must be in PNG, JPG, or PDF format.

        :param bucket_name: The name of the Amazon S3 bucket that contains the
 image.
        :param document_file_name: The name of the document image stored in
 Amazon S3.
        :param sns_topic_arn: The Amazon Resource Name (ARN) of an Amazon SNS
 topic
```

```
                                where the job completion notification is published.
        :param sns_role_arn: The ARN of an AWS Identity and Access Management
(IAM)
                                role that can be assumed by Textract and grants
permission
                                to publish to the Amazon SNS topic.
        :return: The ID of the job.
        """
        try:
            response = self.textract_client.start_document_text_detection(
                DocumentLocation={
                    'S3Object': {'Bucket': bucket_name, 'Name':
document_file_name}},
                NotificationChannel={
                    'SNSTopicArn': sns_topic_arn, 'RoleArn': sns_role_arn})
            job_id = response['JobId']
            logger.info(
                "Started text detection job %s on %s.", job_id,
document_file_name)
        except ClientError:
            logger.exception("Couldn't detect text in %s.", document_file_name)
            raise
        else:
            return job_id
```

- 在 GitHub 上尋找指示和更多程式碼。
- 如需 API 詳細資訊，請參StartDocumentTextDetection在AWSSDK for Python (Boto3) API 參考。

如需完整的清單AWSSDK 開發人員指南和代碼示例，請參閱將 Amazon Textract 與AWSSDK。本主題還包括有關入門信息和以前 SDK 版本的詳細信息。

Amazon Textract 的跨服務範例

以下示例應用程序使用AWS軟件開發工具包將 Amazon Textract 與其他AWS服務。每個範例都包含 GitHub 的鏈接，您可以在其中找到如何設定和執行應用程式的指示。

範例

- 建立 Amazon Textract Explorer 應用程式

- 檢測從圖像中提取的文本中使用AWSSDK

建立 Amazon Textract Explorer 應用程式

下列程式碼範例示範如何透過互動式應用程式探索 Amazon Textract 輸出。

JavaScript

適用於 JavaScript 的 SDK 第 3 版

示範如何使用 AWS SDK for JavaScript 來建置 React 應用程式，該應用程式使用 Amazon Textract 從文件影像擷取資料，並將其顯示在互動式網頁中。此範例會在 Web 瀏覽器中執行，且登入資料需要經過驗證的 Amazon Cognito 身分。它使用 Amazon Simple Storage Service (Amazon S3 進行儲存，對於通知，它會輪詢訂閱 Amazon Simple Notification Service (Amazon SNS)) 主題的 Amazon Simple Queue Service (Amazon SQS) 佇列。

如需完整的原始碼和如何設定及執行的指示，請參閱 GitHub 上的完整範例。

此範例中使用的服務

- Amazon Cognito 身分
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

適用於 Python (Boto3) 的 SDK

示範如何使用 AWS SDK for Python (Boto3)，透過 Amazon Textract 偵測文件影像中的文字、表單和資料表元素。輸入影像和 Amazon Textract 輸出會顯示在 Tkinter 應用程式中，可讓您探索偵測到的元素。

- 將文件影像提交到 Amazon Textract，並探索偵測到元素的輸出。
- 將影像直接傳送至 Amazon Textract 或透過 Amazon Simple Storage Service (Amazon S3) 儲存貯體。
- 使用非同步 API 可以在任務完成時啟動將通知發佈到 Amazon Simple Notification Service (Amazon SNS) 主題的任務。

- 輪詢 Amazon Simple Queue Service (Amazon SQS) 佇列以取得任務完成訊息並顯示結果。

如需完整的原始碼和如何設定及執行的指示，請參閱 GitHub 上的完整範例。

此範例中使用的服務

- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

如需完整的清單AWSSDK 開發人員指南和代碼示例，請參閱將 Amazon Textract 與AWSSDK。本主題還包括有關入門信息和以前 SDK 版本的詳細信息。

檢測從圖像中提取的文本中使用AWSSDK

以下代碼示例說明如何使用 Amazon Comprehend 檢測 Amazon Amazon Textract 從存儲在 Amazon S3 中的圖像中提取的文本中的實體。

Python

適用於 Python (Boto3) 的 SDK

示範如何使用AWS SDK for Python (Boto3)在 Jupyter 筆記本中檢測從圖像中提取的文本中的實體。此範例使用 Amazon Textract 從儲存於 Amazon Simple Simple Simple Simple Simple Simple SSimple Storage Service (Amazon S3) 和 Amazon Comprehend 中的影像中提取文字，以偵測提取文本中的實體。

此示例是 Jupyter 筆記本電腦，必須在可以託管筆記本的環境中運行。如需使用 Amazon SageMaker 執行範例的指示，請參文本和書寫筆記本電腦。

如需完整的原始碼和如何設定及執行的指示，請參閱 GitHub 上的完整範例。

此範例中使用的服務

- Amazon Comprehend
- Amazon S3
- Amazon Textract

如需完整的清單AWSSDK 開發人員指南和代碼示例，請參閱將 Amazon Textract 與AWSSDK。本主題還包括有關入門信息和以前 SDK 版本的詳細信息。

使用亞馬遜 Augmented AI 向 Amazon Textract 輸出添加人工評論

Amazon Augmented AI (Amazon A2I) 是一項機器學習 (ML) 服務，可讓您輕鬆地建立人工檢 ML 分析的工作流程。

Amazon Textract I 集成了 Amazon A2I。您可以使用它將置信度較低的文檔分析結果發送給人工審閱者。

您可以使用 Amazon Textract`AnalyzeDocument`API 從表單和亞馬遜 A2I 控制台中提取數據。您可以指定 Amazon A2I 將預測傳送給審閱者的條件。您可以根據重要表單鍵的置信閾值設置條件。例如，您可以將文檔發送給人工審閱者，如果鍵名稱或其關聯值Jane Doe檢測到的可信度較低。

主題

- Amazon A2I 的核心概念
- 使用 Amazon A2I

Amazon A2I 的核心概念

查看以下術語以熟悉亞馬遜 A2I 的核心概念。

人工檢激活條件

您可以使用 Amazon A2I激活條件，以指定何時將文檔發送給人類進行審查，以及要求工作人員審閱的表格內容。

例如，您可以設置激活條件，以便在檢測到重要密鑰時，Amazon Textract 將表單路由到 Amazon A2I，例如電話號碼。在此示例中，要求人工審閱者查看電話號碼字段和 Amazon Textract 檢測到的值。

您可以使用以下激活條件來指定表單何時發送給人類以供審閱：

- 根據表單鍵可信度分數，觸發特定表單鍵的人工檢閱。人工檢者必須檢這些表單鍵和相關值。
- 在缺少特定表單鍵時觸發人工檢閱。要求人工審閱者識別這些表單鍵和相關值。
- 針對 Amazon Textract 識別且可信度分數在指定範圍內的所有表單鍵，觸發人工檢。

- 隨機傳送表單樣本供人工檢閱。人工檢者必須檢 Amazon Textract 檢測到的所有表單鍵和值。

當您的啟用條件取決於表單鍵可信度分數時，您可以使用兩種類型的預測可信度閾值來觸發人工檢：

- 識別可信度— 在表單內檢測到的鍵值對的可信度分數。
- 資格可信度— 表單中鍵值對中包含的文字的可信度分數。

如果您指定置信閾值，Amazon A2I 將僅將屬於閾值範圍內的預測路由給人工審閱者。您可以隨時調整這些閾值，以實現準確性和成本效益之間的正確平衡。這可以幫助您實施審核以定期監控預測準確性。

> **Note**
>
> 您可以使用 Amazon A2I 自定義任務類型進一步自定義將文檔發送給人類進行審核的條件。使用此任務類型，您可以直接在應用程序中指定人工審查的條件。如需相關資訊，請參針對自訂任務類型使用 Amazon 增強型 AI(位於 Amazon SageMaker 開發人員指南中)。

人工審核工作流程（流程定義）

您可以使用人工檢工作流程，也稱為流程定義，指定用於創建人工審核工作流程的資源，並指定激活條件。

您指定的資源包括：

- 具有調用亞馬遜 A2I API 操作權限的 IAM 角色
- 您想要在其中存放人工檢輸出的 Amazon S3 儲存貯體
- 你的人力工作團隊
- 一個工作者任務範本，其中包括幫助工作人員完成審核任務的說明和示例

您還可以使用人工審核工作流指定激活條件。如需詳細資訊，請參閱 人工檢激活條件。

您可以使用單個人工檢工作流程來建立多個人體循環。

您可以在 SageMaker 主控台或使用 SageMaker API 建立人工檢工作流程。如需相關資訊，請參閱 建立人工檢工作流程。

工作人員任務模板

您可以使用工作者任務範本創建用於人工審核任務的工作人員 UI。

工作人員 UI 將顯示您的文檔和工作人員說明。它還提供工作者完成您的任務所使用的工具。

在您建立人工檢工作流程時，您可以使用 SageMaker 主控台來配置工作者任務模板。如需相關資訊，請參閱 建立人工檢工作流程。

工作團隊

一個工作團隊是您向其發送人工審核任務的一組人工。

在您建立人工檢工作流程時，您需要指定單個工作團隊。

藉助 Amazon A2I，您可以使用自己組織內的檢者集區。您還可以訪問由超過 500,000 名獨立承包商組成的員工隊伍，他們已經通過 Amazon Mechanical Turk 執行機器學習任務。另一種選擇是使用 AWS 預先篩選的人力廠商，以確保品質並嚴守安全程序。

Amazon A2I 還為審稿人提供了一個 Web 界面，該界面由他們完成審閱任務所需的所有說明和工具組成。

對於每一種人力 (私有人力、供應商和 Mechanical Turk 其人力)，您可以建立多個工作團隊。您可以在多個人工審核工作流中使用每個工作組。若要了解如何建立人力和工作團隊，請參建立和管理人力(位於 Amazon SageMaker 開發人員指南中)。

> **Important**
>
> 按一下這裡，查看此時涵蓋亞馬遜 Augmented AI 的合規計劃。如果您將 Amazon Augmented AI 與其他 AWS 服務（例如 Amazon Resource Resource 和 Amazon Textract）搭配使用，請注意，Amazon Augmented AI 可能不屬於與其他服務相同的合規計劃的範圍內。您應負責使用 Amazon Augmented AI 的方式，包括理解服務如何處理或儲存客户數據，以及對您數據環境合規的任何影響。您應與 AWS 客户團隊討論工作負載方向與目標；他們可以幫助您評估服務是否適合您提出的使用案例和架構。
> 目前，除了公共和供應商員工案例外，亞馬遜 Augmented AI 符合 PCI 要求。
> 有關亞馬遜 Augmented AI HIPAA 合規性的信息，請單擊這裡。

人體循環

您可以使用人工循環建立人工檢任務。

您將人工審核任務分配給人工團隊上的工作人員。系統要求工作人員查看 Amazon Textract 在您在激活條件中指定的輸入文檔中檢測到的鍵值對。

例如，假設一張圖片下降到百分之五十到六十的信心，它包含一個蘋果。這屬於人工審查的信心閾值範圍內，並將其發送給工作人員。工作人員檢查文檔是否存在蘋果，並將其標記為包含或不包含蘋果。然後，亞馬遜 A2I 將文檔重新發送到工作流程中。

當您撥打 Amazon Textract`AnalyzeDocument`並指定人工審核工作流程（流程定義）和人工循環名稱，則在滿足人工審核工作流程中指定的激活條件時，將創建人工審閱任務。這些任務是使用您在人工審核工作流程中指定的資源創建的。

使用 Amazon A2I

以下步驟可幫助您將亞馬遜 A2I 集成到 Amazon Textract 單頁文檔分析任務中。您可執行下列項目：

1. 使用亞馬遜 A2I 控制台（推薦給新用户）或亞馬遜 A2I API 創建人工審核工作流程。
2. 要分析表單並在必要時包括人工審核，請使用`AnalyzeDocument`操作並指定人工檢工作流程的 Amazon Resource Name (ARN)。回應會告訴您是否需要人工檢。
3. 使用 Amazon A2I 主控台和 API 監控您的人工迴圈。
4. 在發送結果的 Amazon S3 存儲桶中查看人工審核結果。

要設置 SageMaker 筆記本實例並使用示例筆記本，請參閱使用 Amazon Textract 和 Augmented AI 的端對端示範中的Amazon SageMaker 開發人員指南

> **Note**
>
> 本節介紹如何為亞馬遜 A2I、亞馬 Amazon Textract 任務類型創建人工審核工作流程。要進一步自定義亞馬遜 A2I 和 Amazon Textract 集成，您可以使用亞馬遜 A2I 自定義任務類型。使用此選項，您可以提供自定義工作人員任務模板，並指定直接在應用程序中發送文檔以供人工審閱的條件。如需相關資訊，請參閱「」。針對自訂任務類型使用 Amazon 增強型 AI中的Amazon SageMaker 開發人員指南。

主題

- 建立人工檢工作流程
- 分析文檔
- 監控人工迴圈

- 查看輸出數據和工作程序指標

建立人工檢工作流程

您可以使用亞馬遜 A2I 控制台（推薦給新用户）或亞馬遜 A2I 創建人工審核工作流程`CreateFlowDefinition`operation.

主題

- 建立人工檢閱工作流程 (主控台)
- 建立人工檢閱工作流程 (API)

建立人工檢閱工作流程 (主控台)

您可以使用 Amazon S3 中自己的文檔完成此示例，也可以下載此範例文檔並將其放入您的 Amazon S3 儲存貯體中。

確保您的 S3 儲存貯體位於相同的AWS您正在使用 Amazon Textract 的區域。若要建立儲存儲體，請參《》建立儲存貯體中的Amazon Simple Storage Service 主控台用户指南。

> **Note**
>
> 亞馬遜 A2I 控制台嵌入到 SageMaker 控制台中。若要使用主控台，您需要訪問 SageMaker 主控台和建立工作團隊的許可。若要開始使用，您可以使用AmazonSageMakerFullAccessIAM 託管策略，其中包含在 SageMaker 中執行大多數操作所需的所有必要許可。如需詳細資訊，請參閱「」適用於 Amazon SagMaker 的 Identity of Access Management中的Amazon SageMaker 開發人員指南。

主題

- 步驟 1：建立工作團隊（主控台）
- 步驟 2：建立人工檢閱工作流程 (主控台)

步驟 1：建立工作團隊（主控台）

首先，在 Amazon A2I 控制台中創建一個工作組，並將自己添加為工作人員，以便您可以在工作人員門户中預覽人工審核任務，工作團隊成員可以查看分配給他們的不同任務和文檔。

使用工作者電子郵件建立私有人力 (主控台)

1. 開啟位於的 SageMaker 主控台https://console.aws.amazon.com/sagemaker/。
2. 在導覽窗格中的Ground Truth，選擇標識工作人力。
3. 選擇 Private (私有)，然後選擇 Create private team (建立私有團隊)。
4. 選擇 Invite new workers by email (透過電子郵件邀請新的工作者)。
5. 在此示例中，輸入您的電子郵件地址和您希望能夠預覽工作人員門户的任何其他人的電子郵件地址。您可以粘貼或輸入最多 50 個電子郵件地址的清單 (以逗號分隔) 到電子郵件地址框。
6. 輸入一個組織名稱和聯絡人電子郵件。
7. 選擇 Create private team (建立私有團隊)。

如果您將自己添加到私人工作團隊中，您將收到來自no-reply@verificationemail.com，包含登錄信息。使用此電子郵件中的鏈接重置您的密碼並登錄工作線程門户。這是在您調用AnalyzeDocument。

步驟 2：建立人工檢閱工作流程 (主控台)

在此步驟中，您將創建 Amazon Textract 人工審核工作流程。

建立人工檢工作流程 (主控台)

1. 開啟位於的 Amazon A2I 主控台https://console.aws.amazon.com/a2i存取人工審查工作流(憑證已建立！) 頁面上的名稱有些許差異。
2. 選擇創建人工審核工作流。
3. 適用於名稱中，輸入工作流程名稱。
4. 適用於S3 儲存貯體中，選取您希望 Amazon A2I 儲存人工檢任務結果的儲存貯體。如果您未選擇儲存貯體，請將其更改為輸入儲存貯體的名稱。
5. UNDERIAM 角色，選取Create a new role (建立新角色)。將顯示一個窗口，其標題為建立 IAM 角色。使用此窗口可以指定您希望此角色具有訪問權限的 Amazon S3 存儲桶。如果您沒有選擇任何 S3 儲存儲體，指定您在步驟 4 中指定的輸出存儲桶和包含輸入文檔的存儲桶。
6. 適用於任務類型，選擇Textract-鍵值組提取。
7. InAmazon Textract 表單提取-調用人工審查的條件中，指定激活條件。我們建議您在文檔中至少為一個密鑰設置高置信度閾值，以觸發人工審核，以便您可以在工作人員門户中預覽工作人員任務。

   如果使用了本演練中提供的示例文檔，請按如下方式指定激活條件：

a. 選擇根據表單鍵可信度分數或特定表單鍵缺失時，觸發特定表單鍵的人工檢。
b. 適用於金鑰名稱，輸入**Mail Address**。
c. 將識別可信度閾值介於0和99。
d. 將資格可信度閾值介於0和99。
e. 選擇針對 Amazon Textract 識別且可信度分數的所有表單鍵，觸發人工檢。
f. 適用於**identification confidence**中，選擇介於 0 和 90 之間的任意編號。
g. 適用於**qualification confidence**中，選擇介於 0 和 90 之間的任意編號。

如果 Amazon Textract 返回的置信度分數小於 99，則會觸發人工審查郵件地址及其值，或者返回文檔中檢測到的任何鍵值對的置信度小於 90。

8. UNDER工作人員任務模板創建，選取從默認模板創建。
9. 適用於範本名稱中，輸入描述性名稱。
10. 適用於Task description (任務描述)，請添加類似下列的內容：

```
Read the instructions and review the document.
```

11. 適用於工作者選擇私有。
12. 從菜單中選擇您創建的專用團隊。
13. 選取 Create (建立)。

創建人工審閱工作流後，該工作流將顯示在人工審查工作流(憑證已建立！) 頁面上的名稱有些許差異。當狀態是作用中中，複製並保存工作流 ARN。

建立人工檢閱工作流程 (API)

您可以建立人工檢工作流程或流程定義，使用 Amazon A2I`CreateFlowDefinition`operation.

對於此示例，您可以在 Amazon S3 中使用您自己的文檔，也可以下載此範例文檔並將其存放在 S3 儲存貯體中。

確保您的 Amazon S3 儲存貯體位於相同的AWS您計劃用於調用`AnalyzeDocument`。若要建立儲存貯體，請遵循建立儲存貯體中的Amazon Simple Storage Service 主控台用户指南。

先決條件

要使用 Amazon A2I API 創建人工審核工作流程，您必須完成以下先決條件：

- 配置具有調用亞馬遜 A2I 和 Amazon Textract API 操作權限的 IAM 角色。要開始使用，您可以將 AWS 政策、AmazonAugmentedAIFullAccess 和 AmazonTextractFullAccess 附加到 IAM 角色。記錄 IAM 角色 Amazon Resource Name (ARN)，因為稍後需要使用。

  有關使用 Amazon Textract 時的更詳細的權限，請參閱Amazon Textract 身分型政策範例。如需 Amazon A2I，請參「」。Amazon Augmented AI 中的許可和安全性中的Amazon SageMaker 開發人員指南。
- 創建一個私人工作小組並記錄工作組 ARN。如果您是 Amazon A2I 的新用户，請按照步驟 1：建立工作團隊（主控台）。
- 建立工作者任務範本。請遵循建立工作者任務範本以使用亞馬遜 A2I 控制台創建模板。創建模板時，選擇文本形式提取為了模板類型。在模板中，將`s3_arn`與您的文檔的 Amazon S3 ARN 執行個體。添加其他工作人員説明`<full-instructions header="Instructions"></full-instructions>`。

  如果您想要預覽模板，請確保您的 IAM 角色具有啟用工作者任務範本預覽。

  創建模板後，記錄工作人員任務模板 ARN。

您可以使用您在先決條件以配置您的`CreateFlowDefinition`請求。在此請求中，您還可以使用 JSON 格式指定啟用條件。若要了解如何設定啟用條件，請參使用人工迴圈啟用條件 JSON 架構與 Amazon Textract。

建立人工檢工作流程 (適用於 Python 的 AWS 開發套件 (Boto3))

若要使用此範例，請將*red*文本與您的規格和資源。

首先，使用以下代碼將激活條件編碼為 JSON 對象。如果 Amazon Textract 返回的置信度分數小於 99，則會觸發人工審查郵件地址及其值，或者返回文檔中檢測到的任何鍵值對的置信度小於 90。如果您使用本示例中提供的示例文檔，則這些激活條件會創建人工審核任務。

```
import json

humanLoopActivationConditions = json.dumps("{
                "Conditions": [
                    {
                        "ConditionType": "ImportantFormKeyConfidenceCheck",
                        "ConditionParameters": {
                            "ImportantFormKey": "Mail Address",
                            "KeyValueBlockConfidenceLessThan": 99,
                            "WordBlockConfidenceLessThan": 99
```

```
                    }
                },
                {
                    "ConditionType": "ImportantFormKeyConfidenceCheck",
                    "ConditionParameters": {
                        "ImportantFormKey": "*",
                        "KeyValueBlockConfidenceLessThan": 90,
                        "WordBlockConfidenceLessThan": 90
                    }
                }
            ]
        }"
)
```

使用humanLoopActivationConditions配置create_flow_definition請求。下面的示例使用 SDK for Python (Boto3) 調用create_flow_definition（us-west-2 AWS 區域）。它指定使用私有工作團隊。

```
response = client.create_flow_definition(
    FlowDefinitionName='string',
    HumanLoopRequestSource={
         'AwsManagedHumanLoopRequestSource': "AWS/Textract/AnalyzeDocument/Forms/V1"
    },
    HumanLoopActivationConfig={
        'HumanLoopActivationConditionsConfig': {
            'HumanLoopActivationConditions': humanLoopActivationConditions
        }
    },
    HumanLoopConfig={
        'WorkteamArn': "arn:aws:sagemaker:us-west-2:111122223333:workteam/private-
crowd/work-team-name",
        'HumanTaskUiArn': "arn:aws:sagemaker:us-west-2:111122223333:human-task-
ui/worker-task-template-name",
        'TaskTitle': "Add a task title",
        'TaskDescription': "Describe your task",
        'TaskCount': 1,
        'TaskAvailabilityLifetimeInSeconds': 3600,
        'TaskTimeLimitInSeconds': 86400,
        'TaskKeywords': ["Document Review", "Content Review"]
        }
    },
    OutputConfig={
        'S3OutputPath': "s3://DOC-EXAMPLE-BUCKET/prefix/",
```

```
    },
    RoleArn="arn:aws:iam::111122223333:role/role-name"
)
```

分析文檔

要將亞馬遜 A2I 整合到 Amazon Textract 文檔分析工作流程中，請配置HumanLoopConfig中的AnalyzeDocumentoperation.

InHumanLoopConfig中，您可以指定人工檢工作流程 (流程定義)FlowDefinitionArn，並給你的人類循環HumanLoopName。

Analyze the Document (AWS SDK for Python (Boto3))

下面的示例使用 SDK for Python (Boto3) 調用analyze_document在 us-west-2. 替換##, ##文本與您的資源。如需詳細資訊，請參閱「」分析文檔中的AWS適用於 Python (Boto) 的開發套件。

```
client.analyze_document(Document={'S3Object': {"Bucket": "DOC-EXAMPLE-BUCKET",
 "Name": "document-name.png"}},
        HumanLoopConfig={"FlowDefinitionArn":"arn:aws:sagemaker:us-
west-2:111122223333:flow-definition/flow-definition-name",
                         "HumanLoopName":"human-loop-name",
                         "DataAttributes":{"ContentClassifiers":
["FreeOfPersonallyIdentifiableInformation"|"FreeOfAdultContent",]}},
        FeatureTypes=["FORMS"])
```

Analyze the Document (AWS CLI)

下列範例使用AWS要調用的 CLIanalyze_document。這些範例與AWSCLI 版本 2. 第一個是簡寫語法，第二個是 JSON 語法。如需詳細資訊，請參閱「」分析文檔中的AWS CLI命令參考。

```
aws textract analyze-document \
     --document '{"S3Object":{"Bucket":"bucket_name","Name":"file_name"}}' \
     --human-loop-config
 HumanLoopName="test",FlowDefinitionArn="arn:aws:sagemaker:eu-west-1:xyz:flow-
definition/
hl_name",DataAttributes='{ContentClassifiers=["FreeOfPersonallyIdentifiableInformation","Fre
     --feature-types '["FORMS"]'
```

```
aws textract analyze-document \
     --document '{"S3Object":{"Bucket":"bucket_name","Name":"file_name"}}' \
     --human-loop-config \
          '{"HumanLoopName":"test","FlowDefinitionArn":"arn:aws:sagemaker:eu-
west-1:xyz:flow-definition/hl_name","DataAttributes": {"ContentClassifiers":
["FreeOfPersonallyIdentifiableInformation","FreeOfAdultContent"]}}' \
     --feature-types '["FORMS"]'
```

> **Note**
>
> 避免在 —human-loop config 參數中使用白色空格，因為這可能會導致代碼的處理問題。

對此請求的響應包含人類循環活動產出，它指示是否創建了人類循環，如果是，為什麼。如果創建了人類循環，則此對象還包含HumanLoopArn。

如需使用AnalyzeDocument操作，請參閱使用 Amazon Textract 分析文檔文本。

監控人工迴圈

您可以使用 Amazon A2I 控制台和 API 查看有關人工循環的詳細信息，並在出現錯誤時停止活動的人工循環。

查看人工迴圈詳情

您可以在 Amazon A2I 控制台中查看人工循環狀態，並使用Amazon A2I 運行時 API。

查找有關人類循環（控制台）的詳細信息

1. 開啟位於的 Amazon A2I 主控台https://console.aws.amazon.com/a2i存取人工審查工作流(憑證已建立！) 頁面上的名稱有些許差異。
2. 選擇您也用來配置的人工檢工作流程HumanLoopConfig在AnalyzeDocument。
3. 在 中人體循環區段中，選取您想要檢視詳細資訊的人工迴圈。

要查找有關人類循環 (API) 的詳細信息：

使用 Amazon A2IDescribeHumanLoopoperation. 指定用於調用的人類循環名稱AnalyzeDocument。

下列適用於 Python 的開發套件 (Boto3) 範例調用describe_human_loop。

```
response = client.describe_human_loop(HumanLoopName="human-loop-name")
```

停止人工迴圈

人工循環開始後，您可以使用 Amazon A2I 控制台和 API 停止它。

停止你的人類循環（控制台）

1. 開啟位於的 Amazon A2I 主控台https://console.aws.amazon.com/a2i存取人工審查工作流(憑證已建立！) 頁面上的名稱有些許差異。
2. 選擇您用來配置的人工檢工作流程HumanLoopConfig中的AnalyzeDocumentoperation.
3. 在 中人體循環部分中，選擇要停止的人工迴圈。
4. 選擇 Stop (停止)。

停止你的人類循環（API）

使用 Amazon A2IStopHumanLoopoperation. 指定用來調用的人工循環的名稱AnalyzeDocument。

下列 SDK for Python (Boto3) 範例調用stop_human_loop。

```
response = client.stop_human_loop(HumanLoopName="human-loop-name")
```

查看輸出數據和工作程序指標

當工作人員完成人工審核任務時，Amazon A2I 會將您的輸出數據存儲在您在人工審核工作流程中指定的 Amazon S3 存儲桶中。

如果您使用私有工作人員，則輸出數據包含可用於跟蹤單個工作人員活動的工作人員元數據。

在 Amazon S3 中查找輸出資料

Amazon A2I 使用您的人工審核工作流程名稱作為文件名稱的前綴，該文件存儲使用該人工審核工作流程創建的人工循環的輸出數據。

人類循環輸出的路徑使用以下模式，其中*YYYY*/*MM*/*DD*/*hh*/*mm*/*ss*表示帶有年份的人循環創建日期（YYYY), 月份 (MM) 和天 (DD）和創建時間與小時（hh), 分鐘 (mm) 和第二個 (ss。

```
s3://output-bucket-specified-in-flow-definition/flow-definition-
name/YYYY/MM/DD/hh/mm/ss/human-loop-name/output.json
```

要查看人工循環的輸出，請使用 Amazon A2I 控制台。

查看人類循環輸出

1. 開啟位於的 Amazon A2I 主控台https://console.aws.amazon.com/a2i存取人工審查工作流(憑證已建立！) 頁面上的名稱有些許差異。
2. 選擇您用來配置的人工檢工作流程HumanLoopConfig在AnalyzeDocument。
3. 在 中人體循環部分中，選擇要查看其輸出的人類循環。
4. UNDER輸出位置中，選擇指向輸出資料的鏈接。

跟蹤私人工人活動

當您將私人員工作人員用於人工審核任務時，輸出數據包括有關完成審核的工作人員的以下信息：

- workerId。
- 在 workerMetadata 中：
  - identityProviderType— 用來管理私有人力的服務。
  - issuer— Amazon Cognito 用户池或 OIDC 身份提供商 (IdP) 發行者與指派給此人工檢任務的工作團隊相關聯。
  - sub— 指向工作者的唯一標識符。如果您使用 Amazon Cognito 建立人力，您可以使用 Amazon Cognito 檢索有關此工作者的詳細資訊 (例如姓名或用户名)。若要瞭解方法，請參《》管理及搜尋使用者帳戶在Amazon Cognito 開發人員指南。

以下是您是否使用 Amazon Cognito 創建私人員工隊伍時可能會看到的輸出示例。

```
"workerId": "a12b3cdefg4h5i67",
            "workerMetadata": {
                "identityData": {
                    "identityProviderType": "Cognito",
                    "issuer": "https://cognito-idp.aws-region.amazonaws.com/aws-
region_123456789",
                    "sub": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee"
```

以下是您是否使用自己的 OIDC IdP 創建私人員工隊伍時可能會看到的輸出示例：

```
"workerId": "a12b3cdefg4h5i67",
            "workerMetadata": {
                "identityData": {
                    "identityProviderType": "Oidc",
                    "issuer": "https://example-oidc-ipd.com/adfs",
                    "sub": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee"
```

若要了解如何使用私有工作者，請參使用私有人力中的Amazon SageMaker 開發人員指南。

Amazon Ttext 的安全性

雲端安全是 AWS 最重視的一環。身為 AWS 的客戶，您將能從資料中心和網路架構中獲益，這些都是專為最重視安全的組織而設計的。

請使用下列主題來了解如何保護您的 Amazon Textract 資源。

主題

- Amazon Textract 的資料保護
- Amazon Textract 的 Identity to Access Management
- 記錄和監控
- 使用 記錄 Amazon Textract API 呼叫AWS CloudTrail
- Amazon Textract 的合規驗證
- Amazon Textract 的恢復能力
- Amazon Textract 的基礎設施安全
- Amazon Textract 中的組態與漏洞分析
- Amazon Textract 和界面 VPC 端點 (AWS PrivateLink)

Amazon Textract 的資料保護

Amazon Textract 符合AWS 共同責任模型，包含適用於資料保護的法規和指導方針。AWS負責保護全球基礎設施，該基礎設施會運行所有AWS服務。AWS維護控制在此基礎設施上託管的資料，包括處理客戶內容和個人資料的安全配置控制。AWS客戶和APN合作夥伴，作為資料控制器或資料處理器，負責它們放置在AWS雲端。

為了保護資料，我們建議您保護 AWS 帳戶登入資料，並使用 AWS Identity and Access Management (IAM) 來設定個別使用者帳戶。這樣做可確保每個使用者都只會獲得完成其任務所必須的許可。我們也建議您採用下列方式保護資料：

- 每個帳戶都使用多重要素驗證 (MFA)。
- 使用 SSL/TLS 與 AWS 資源通訊。
- 使用 AWS CloudTrail 設定 API 和使用者活動記錄。
- 使用 AWS 加密解決方案，以及 AWS 服務內的所有預設安全控制。

- 使用進階的受管安全服務 (例如 Amazon Macie)，協助探索和保護儲存在 Amazon S3 的個人資料。

我們強烈建議您絕對不要將客戶帳戶號碼等敏感的識別資訊，放在自由格式的欄位中，例如Name (名稱) 欄位。這包括當您使用 Amazon Textract 或其他AWS服務使用控制台、API、AWS CLI, 或AWS開發套件。您在 Amazon Textract 或其他服務中輸入的任何資料都可能選入診斷日誌中。當您提供外部伺服器的 URL 時，請勿在驗證您對該伺服器請求的 URL 中包含憑證資訊。

如需關於資料保護的詳細資訊，請參閱 AWS 安全部落格上的 AWS 共同責任模型和歐盟《一般資料保護規範》(GDPR) 部落格文章。

Amazon Ttext 的加密

資料加密是指在傳輸中和靜態下保護資料。您可以使用 Amazon S3 託管密鑰或AWS KMS key，以及在運輸途中的標準傳輸層安全性。

靜態加密

在 Amazon Textract 中加密數據的主要方法是服務器端加密。從 Amazon S3 存儲桶傳遞的輸入文檔由 Amazon S3 加密，並在您訪問這些文檔時進行解密。只要您驗證要求並具備存取許可，存取加密物件或未加密物件的方式並無不同。例如，如果您使用預先簽章的 URL 來分享物件，加密物件與未加密物件的 URL 運作方式會相同。此外，當您列出儲存貯體中的物件時，`ListAPI` 會返回所有物件的清單，無論這些物件是否已加密。

Amazon Textract 使用兩種互相排斥的伺服器端加密方法。

伺服器端加密搭配 Amazon S3 受管金鑰 (SSE-S3)

使用 Amazon S3 受管金鑰 (SSE-S3) 的服務器端加密搭配使用唯一金鑰來加密每個物件。此方法使用定期輪換的主要金鑰自行加密金鑰，提供額外的防護。Amazon S3 伺服器端加密使用目前最強大的其中一種區塊加密法 (256 位元進階加密標準 (AES-256))，加密您的資料。如需詳細資訊，請參閱使用伺服器端加密與 Amazon S3 受管加密金鑰 (SSE-S3) 保護資料。

伺服器端加密搭配存放在 AWS Key Management Service (SSE-KMS) 中的 KMS 金鑰

「伺服器端加密搭配存放在 AWS Key Management Service (SSE-KMS) 的 KMS 金鑰」與 SSE-S3 相似，但有一些額外優點，且使用此服務需支付一些額外費用。使用 KMS 金鑰需要個別的許可。KMS 金鑰可提供多一層保護，防止他人在未經授權的情況下存取您在 Amazon S3 中的物件。SSE-KMS 也可以提供稽核線索，顯示使用您 KMS 金鑰的人員及使用的時間。此外，您可以創建和管理 KMS 密鑰，或使用AWS 受管金鑰專屬於您、您的服務和您的區域。如需詳細資訊，請參閱「」使用伺服器端加密保護資料使用儲存在 AWS Key Management Service (SSE-KMS) 中的 KMS 金鑰。

傳輸中加密

Amazon Textract 使用傳輸層安全性 (TLS) 來加密服務與代理程式之間發送的資料。此外，Amazon Textract 使用 VPC 終端節點在 Amazon Textract 文檔時使用的各種微服務之間發送數據。

網際網路流量隱私權

Amazon Textract 僅通過 HTTPS 終端節點進行通信，這些終端節點在 Amazon Textract 支持的所有地區都支持

Amazon Textract 的 Identity to Access Management

AWS Identity and Access Management (IAM) 是一種 AWS 服務，讓管理員能夠安全地控制對 AWS 資源的存取。IAM 管理員控制可認證(簽名) 和授權（有權限）使用 Amazon Textract 資源。IAM 是一種您可以免費使用的 AWS 服務。

主題

- 對象
- 使用身分來驗證
- 使用政策管理存取權
- Amazon Textract 如何搭配 IAM 運作
- Amazon Textract 身分型政策範例
- 針對 Amazon Textract Identity to Access 進行故

對象

您使用的方式AWS Identity and Access Management(IAM) 會不同，取決於您在 Amazon Textract 中所執行的工作。

服務用户— 若您使用 Amazon Textract 來執行任務，您的管理員可以提供您需要的登入資料和許可。隨著您為了執行作業而使用的 Amazon Textract 功能數量變多，您可能會需要額外的許可。了解存取的管理方式可協助您向管理員請求正確的許可。若您無法存取 Amazon Textract 中的某項功能，請參針對 Amazon Textract Identity to Access 進行故。

服務管理員— 若您在公司負責管理 Amazon Textract 資源，您應該具備服務使用的完整存取權限。您的任務是要判斷員工應存取哪些 Amazon Textract 功能和資源。您接著必須將請求提交給您的 IAM 管

理員，來變更您服務使用者的許可。檢閱此頁面上的資訊，了解 IAM 的基本概念。若要進一步了解貴公司可搭配 Amazon Textract 使用 IAM 的方式，請參Amazon Textract 如何搭配 IAM 運作。

IAM 管理員— 若您是 IAM 管理員，建議您掌握如何撰寫政策來管理 Amazon Textract 存取權的詳細資訊。若要檢視您可以在 IAM 中使用的 Amazon Textract 身分型政策範例，請參Amazon Textract 身分型政策範例。

使用身分來驗證

身分驗證是使用身分憑證登入 AWS 的方式。若需使用 AWS Management Console 登入的詳細資訊，請參閱《IAM 使用者指南》中的以 IAM 使用者或根使用者身分登入 AWS Management Console。

您必須以 AWS 帳戶 根使用者身分、IAM 使用者身分，或使用 IAM 角色進行驗證 (登入至 AWS)。您也可以使用貴公司的單一登入身分驗證，甚至使用 Google 或 Facebook 進行登入。在上述案例中，您的管理員會使用 IAM 角色預先設定聯合身分。當您使用來自其他公司的身分驗證來存取 AWS 時，您是間接地擔任角色。

若要直接登入 AWS Management Console，請使用您的根使用者電子郵件地址或您的 IAM 使用者名稱密碼。您可以使用您的根使用者或 IAM 使用者存取金鑰，透過程式設計程式的方式存取 AWS。AWS 提供開發套件和命令列工具，以加密的方式使用您的憑證簽署您的請求。如果您不使用 AWS 工具，您必須自行簽署請求。請使用 Signature 第 4 版來執行此作業，它是針對傳入 API 請求進行身分驗證的通訊協定。如需驗證請求的詳細資訊，請參閱《AWS 一般參考》中的 Signature 第 4 版簽署程序。

無論您使用何種身分驗證方法，您可能還需要提供額外的安全性資訊。例如，AWS 建議您使用多重要素驗證 (MFA) 來提高帳戶的安全。若要進一步了解，請參閱《IAM 使用者指南》中的在 AWS 中使用多重要素驗證 (MFA)。

AWS 帳戶 根使用者

如果是首次建立 AWS 帳戶，您會先有單一的登入身分，可以完整存取帳戶中所有 AWS 服務與資源。此身分稱為 AWS 帳戶 根使用者，使用建立帳戶時所使用的電子郵件地址和密碼即可登入並存取。強烈建議您不要以根使用者處理日常作業，即使是管理作業。反之，請遵循僅將根使用者用來建立您第一個 IAM 使用者的最佳實務。接著請妥善鎖定根使用者憑證，只用來執行少數的帳戶與服務管理作業。

IAM 使用者和群組

IAM 使用者是您 AWS 帳戶 中的一種身分，具備單一人員或應用程式的特定許可。IAM 使用者可有長期憑證 (例如，使用者名稱和密碼或一組存取金鑰)。若要了解如何產生存取金鑰，請參閱《IAM 使用者指南》中的管理 IAM 使用者的存取金鑰。當您產生 IAM 使用者的存取金鑰時，請確認您已檢視且安全地儲存金鑰對。您在這之後便無法復原私密存取金鑰。屆時您必須改為產生新的存取金鑰對。

IAM 群組是一種指定 IAM 使用者集合的身分。您無法以群組身分登入。您可以使用群組來一次為多名使用者指定許可。群組可讓管理大量使用者許可的過程變得更為容易。例如，您可以擁有一個名為 IAMAdmins 的群組，並給予該群組管理 IAM 資源的許可。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供臨時憑證。若要進一步了解，請參閱《IAM 使用者指南》中的建立 IAM 使用者 (而非角色) 的時機。

IAM 角色

IAM 角色是您 AWS 帳戶 中的一種身分，具備特定許可。它類似 IAM 使用者，但不與特定的人員相關聯。您可以在 AWS Management Console 中透過切換角色來暫時取得 IAM 角色。您可以透過呼叫 AWS CLI 或 AWS API 操作，或是使用自訂 URL 來取得角色。如需使用角色的方法詳細資訊，請參閱《IAM 使用者指南》中的使用 IAM 角色。

使用臨時憑證的 IAM 角色在下列情況中非常有用：

- 暫時 IAM 使用者許可：使用者可以擔任 IAM 角色來暫時針對特定任務採用不同的許可。
- 聯合身分使用者存取：並非建立 IAM 使用者，而是使用來自 AWS Directory Service、您的企業使用者目錄或 Web 身分供應商現有的使用者身分。這些稱為聯合身分使用者。透過身分供應商來請求存取時，AWS 會指派角色給聯合身分使用者。如需聯合身分使用者的詳細資訊，請參閱《IAM 使用者指南》中的聯合身分使用者和角色。
- 跨帳戶存取：您可以使用 IAM 角色，允許不同帳戶中的某人 (信任的委託人) 存取您帳戶的資源。角色是授予跨帳戶存取的主要方式。但是，針對某些 AWS 服務，您可以將政策直接連接到資源 (而非使用角色作為代理)。若要了解跨帳戶存取角色和資源型政策間的差異，請參閱《IAM 使用者指南》中的 IAM 角色與資源類型政策的差異。
- 跨服務存取：有些 AWS 服務會使用其他 AWS 服務中的功能。例如，當您在服務中進行呼叫時，該服務通常會在 Amazon EC2 中執行應用程式或將物件存放在 Amazon S3 中。服務可能會使用呼叫委託人的許可、使用服務角色或使用服務連結角色來執行此作業。
  - 委託人許可：當您使用 IAM 使用者或角色在 AWS 中執行動作時，您會被視為委託人。政策能將許可授予委託人。當您使用某些服務時，您可能會執行一個動作，然後在不同的服務中觸發另一個動作。在此情況下，您必須具有執行這兩個動作的許可。若要檢視某個動作是否需要政策中的其他相依動作，請參Amazon Textract 的動作、資源和條件金鑰中的服務授權參考。
  - 服務角色 – 服務角色是服務擔任的 IAM 角色，可代表您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱《IAM 使用者指南》中的建立角色以委派許可給 AWS 服務。

- 服務連結角色：服務連結角色是一種連結到 AWS 服務的服務角色類型。服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的 IAM 帳戶中，並由該服務所擁有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。
- 在 Amazon EC2 上執行的應用程式：針對在 EC2 執行個體上執行並提出 AWS CLI 和 AWS API 請求的應用程式，您可以使用 IAM 角色來管理臨時憑證。這是在 EC2 執行個體內存放存取金鑰的較好方式。若要指派 AWS 角色給 EC2 執行個體並提供其所有應用程式使用，您可以建立連接到執行個體的執行個體設定檔。執行個體設定檔包含該角色，並且可讓 EC2 執行個體上執行的程式取得臨時憑證。如需詳細資訊，請參閱《IAM 使用者指南》中的利用 IAM 角色來授予許可給 Amazon EC2 執行個體上執行的應用程式。

若要了解是否要使用 IAM 角色或 IAM 使用者，請參閱《IAM 使用者指南》中的建立 IAM 角色 (而非使用者) 的時機。

使用政策管理存取權

您可以透過建立政策並將其連接到 IAM 身分或 AWS 資源，在 AWS 中控制存取。政策為 AWS 中的一個物件，當與身分或資源相關聯時，會定義它們的許可。您可以根使用者或 IAM 使用者的身分登入，或者可以擔任 IAM 角色。當您接著提出請求時，AWS 會評估相關身分型或資源型政策。政策中的許可，決定是否允許或拒絕請求。大部分政策以 JSON 文件形式存放在 AWS 中。如需 JSON 政策文件結構和內容的詳細資訊，請參閱《IAM 使用者指南》中的 JSON 政策概觀。

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

每個 IAM 實體 (使用者或角色) 在開始時都沒有許可。換句話說，根據預設，使用者無法執行任何作業，甚至也無法變更他們自己的密碼。若要授予使用者執行動作的許可，管理員必須將許可政策連接到使用者。或者，管理員可以將使用者新增到具備預定許可的群組。管理員將許可給予群組時，該群組中的所有使用者都會獲得那些許可。

IAM 政策定義該動作的許可，無論您使用何種方法來執行操作。例如，假設您有一個允許 `iam:GetRole` 動作的政策。具備該政策的使用者便可以從 AWS Management Console、AWS CLI 或 AWS API 取得角色資訊。

身分型政策

身分型政策是可以附加到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要了解如何建立身分類型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

身分型政策可進一步分類成內嵌政策或受管政策。內嵌政策會直接內嵌到單一使用者、群組或角色。受管政策則是獨立的政策，您可以將這些政策連接到 AWS 帳戶 中的多個使用者、群組和角色。受管政策包含 AWS 受管政策和客戶受管政策。若要了解如何在受管政策及內嵌政策間選擇，請參閱《IAM 使用者指南》中的在受管政策和內嵌政策間選擇。

資源型政策

資源型政策是連接到資源的 JSON 政策文件。資源型政策的最常見範例是 Amazon S3 儲存貯體政策和 IAM 角色信任政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取。對於附加政策的資源，政策會定義指定的委託人可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定委託人。委託人可以包括帳戶、使用者、角色、聯合身分使用者或 AWS 服務。

資源型政策是位於該服務中的內嵌政策。您無法在以資源為基礎的政策中使用來自 IAM 的 AWS 受管政策。

存取控制清單 (ACL)

存取控制清單 (ACL) 可控制哪些委託人 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於以資源為基礎的政策，但它們不使用 JSON 政策文件格式。

Amazon S3、AWS WAF 和 Amazon VPC 是支援 ACL 的服務範例。若要進一步了解 ACL，請參閱《Amazon Simple Storage Service 開發人員指南》中的存取控制清單 (ACL) 概觀。

其他政策類型

AWS 支援其他較少見的政策類型。這些政策類型可設定較常見政策類型授予您的最大許可。

- 許可界限：許可界限是一種進階功能，可供您設定身分型政策能授予 IAM 實體 (IAM 使用者或角色) 的最大許可。您可以為實體設定許可界限。所產生的許可會是實體的身分型政策和其許可界限的交集。會在 `Principal` 欄位中指定使用者或角色的資源型政策則不會受到許可界限限制。所有這類政策中的明確拒絕都會覆寫該允許。如需許可邊界的詳細資訊，請參閱《IAM 使用者指南》中的 IAM 實體許可邊界。
- 服務控制政策 (SCP)：SCP 是 JSON 政策，可指定 AWS Organizations 中組織或組織單位 (OU) 的最大許可。AWS Organizations 服務可用來分組和集中管理您企業所擁有的多個 AWS 帳戶。若您啟用組織中的所有功能，您可以將服務控制政策 (SCP) 套用到任何或所有帳戶。SCP 會限制成員帳戶中的實體許可，包括每個 AWS 帳戶 的根使用者。如需 Organizations 和 SCP 的詳細資訊，請參閱《AWS Organizations 使用者指南》中的 SCP 運作方式。
- 工作階段政策：工作階段政策是一種進階政策，您可以在透過編寫程式的方式建立角色或聯合身分使用者的暫時工作階段時，作為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工

作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需詳細資訊，請參閱《IAM 使用者指南》中的工作階段政策。

多種政策類型

將多種政策類型套用到請求時，其結果形成的許可會更為複雜、更加難以理解。若要了解 AWS 在涉及多種政策類型時如何判斷是否允許一項請求，請參閱《IAM 使用者指南》中的政策評估邏輯。

Amazon Textract 如何搭配 IAM 運作

在您使用 IAM 管理對 Amazon Textract 的存取權之前，您應該瞭解哪些 IAM 功能可以與 Amazon Textract 搭配使用。若要了解 Amazon Textract 和其他AWS服務可搭配 IAM 運作，請參AWS可搭配 IAM 運作的服務中的IAM User Guide。

主題

- Amazon Textract 身分型政策
- Amazon Textract 資源型政策
- 以 Amazon Textract 標籤為基礎的授權
- Amazon Textract IAM 角色

Amazon Textract 身分型政策

使用 IAM 身分型政策，您可以指定允許或拒絕的動作和資源，以及在何種條件下允許或拒絕動作。Amazon Textract 支援特定動作、資源和條件金鑰。若要了解您在 JSON 政策中使用的所有元素，請參閱《IAM 使用者指南》中的 JSON 政策元素參考。

動作

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

JSON 政策的 `Action` 元素描述您可以用來允許或拒絕政策中存取的動作。政策動作的名稱通常會和相關聯的 AWS API 操作相同。有一些例外狀況，例如沒有相符的 API 作業的僅限許可動作。也有一些作業需要政策中的多個動作。這些額外的動作稱為相依動作。

政策會使用動作來授予執行相關聯操作的許可。

Amazon Textract 中的異步操作需要授予兩個操作權限，一個用於「開始」操作，另一個用於獲取操作。此外，如果您使用 Amazon S3 存儲桶傳遞文檔，則需要授予您的賬户讀取訪問權限。

在 Amazon Textract 中，所有策略操作都以以下幾點開始：`textract:`。 例如，若要授予某人使用 Amazon Textract 操作的許可`AnalyzeDocument`操作時，您可以將`textract:AnalyzeDocument`行動在他們的政策。政策陳述式必須包含 `Action` 或 `NotAction` 元素。Amazon Textract 會定義自己的一組動作，描述您可以使用此服務執行的任務。

若要在單一陳述式中指定多個動作，請用逗號分隔，如下所示。

```
"Action": [
      "textract:action1",
      "textract:action2"
```

您也可以使用萬用字元 (*) 來指定多個動作。例如，若要指定開頭是 `Describe` 文字的所有動作，請包含以下動作：

```
"Action": "textract:Describe*"
```

如需 Amazon Textract 殊操作的清單，請參Amazon Textract 定義的動作中的IAM User Guide。

資源

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

`Resource` JSON 政策元素可指定要套用動作的物件。陳述式必須包含 `Resource` 或 `NotResource` 元素。最佳實務是使用其 Amazon 資源名稱 (ARN) 來指定資源。您可以針對支援特定資源類型的動作 (稱為資源層級許可) 來這麼做。

對於不支援資源層級許可的動作 (例如列出作業)，請使用萬用字元 (*) 來表示陳述式適用於所有資源。

```
"Resource": "*"
```

Amazon Textract 不支援在政策中指定資源 ARN。

條件索引鍵

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

`Condition` 元素 (或 `Condition` 區塊) 可讓您指定使陳述式生效的條件。`Condition` 元素是選用項目。您可以建立使用條件運算子的條件表達式 (例如等於或小於)，來比對政策中的條件和請求中的值。

若您在陳述式中指定多個 Condition 元素，或是在單一 Condition 元素中指定多個金鑰，AWS 會使用邏輯 AND 操作評估他們。若您為單一條件索引鍵指定多個值，AWS 會使用邏輯 OR 操作評估條件。必須符合所有條件，才會授予陳述式的許可。

您也可以在指定條件時使用預留位置變數。例如，您可以只在使用者使用其 IAM 使用者名稱標記時，將存取資源的許可授予該 IAM 使用者。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 政策元素：變數和標籤。

AWS 支援全域條件索引鍵和服務特定的條件索引鍵。若要查看 AWS 全域條件索引鍵，請參閱《IAM 使用者指南》中的 AWS 全域條件內容索引鍵。

Amazon Textract 不提供任何服務專用條件金鑰，但它支援使用一些全域條件金鑰。如需所有AWS全域條件索引鍵，請參AWS全球條件內容金鑰中的IAM User Guide。

範例

若要檢視 Amazon Textract 身分類型政策的範例，請參Amazon Textract 身分型政策範例。

Amazon Textract 資源型政策

Amazon Textract 不支援資源類型政策。

以 Amazon Textract 標籤為基礎的授權

Amazon Textract 不支援根據標籤來標記資源或控制存取。

Amazon Textract IAM 角色

IAM 角色是您 AWS 帳戶中具備特定許可的實體。

搭配使用暫時登入資料與 Amazon Textract

您可以搭配聯合使用臨時憑證、擔任 IAM 角色，或是擔任跨帳戶角色。您取得臨時安全憑證的方式是透過呼叫 AWS STS API 操作 (例如，AssumeRole 或 GetFederationToken)。

Amazon Textract 支援使用臨時登入資料。

服務連結角色

服務連結角色可讓 AWS 服務存取其他服務中的資源，以代您完成動作。服務連結角色會顯示在您的 IAM 帳戶中，並由該服務所擁有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

Amazon Textract 不支援服務連結角色。

> **Note**
>
> Amazon Textract 不支援服務連結角色，因此不支援 AWS 服務委託人。如需服務主體的詳細資訊，請參AWS 服務委託人中的IAM User Guide

服務角色

此功能可讓服務代表您擔任服務角色。此角色可讓服務存取其他服務中的資源，以代表您完成動作。服務角色會出現在您的 IAM 帳戶中，且由該帳戶所擁有。這表示 IAM 管理員可以變更此角色的許可。不過，這樣可能會破壞此服務的功能。

Amazon Textract 支援服務角色。

Amazon Textract 身分型政策範例

在默認情況下，IAM 使用者和角色不具備建立或修改 Amazon Textract 資源的許可。他們也無法使用 AWS Management Console、AWS CLI 或 AWS API 執行任務。IAM 管理員必須建立 IAM 政策，授予使用者和角色在指定資源上執行特定 API 操作的所需許可。管理員接著必須將這些政策連接至需要這些許可的 IAM 使用者或群組。

若要了解如何使用這些範例 JSON 政策文件建立 IAM 身分型政策，請參閱《IAM 使用者指南》中的在 JSON 標籤上建立政策。

主題

- 政策最佳實務
- 允許使用者檢視自己的許可
- 允許訪問 Amazon Textract 中的同步操作
- 允許訪問 Amazon Textract 中的異步操作

政策最佳實務

身分型政策相當強大。他們可以判斷您帳號中的某個人員是否可以建立、存取或刪除 Amazon Textract 資源。這些動作可能會讓您的 AWS 帳戶 產生費用。當您建立或編輯身分型政策時，請遵循下列準則及建議事項：

- 開始使用AWS受管政策— 要快速開始使用 Amazon Textract，請使用AWS受管政策，以連結您的員工所需許可。這些政策已在您的帳戶中提供，並由 AWS 維護和更新。如需詳細資訊，請參閱《IAM 使用者指南》中的開始搭配 AWS 受管政策使用許可。
- 授予最低權限：當您建立自訂政策時，請只授予執行任務所需要的許可。以最小一組許可開始，然後依需要授予額外的許可。這比一開始使用太寬鬆的許可，稍後再嘗試將他們限縮更為安全。如需詳細資訊，請參閱《IAM 使用者指南》中的授予最低權限。
- 為敏感操作啟用 MFA：為了增加安全，請要求 IAM 使用者使用多重要素驗證 (MFA) 存取敏感資源或 API 操作。如需詳細資訊，請參閱《IAM 使用者指南》中的在 AWS 中使用多重要素驗證 (MFA)。
- 使用政策條件以增加安全：在切實可行的範圍中，請定義您身分類型政策允許存取資源的條件。例如，您可以撰寫條件，指定請求必須來自一定的允許 IP 地址範圍。您也可以撰寫條件，只在指定的日期或時間範圍內允許請求，或是要求使用 SSL 或 MFA。如需詳細資訊，請參閱「」IAM JSON 政策元素：Condition中的IAM User Guide。

允許使用者檢視自己的許可

此範例會示範如何建立政策，允許 IAM 使用者檢視連接到他們使用者身分的內嵌及受管政策。此政策包含在主控台上，或是使用 AWS CLI 或 AWS API 透過編寫程式的方式完成此動作的許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
```

```
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

允許訪問 Amazon Textract 中的同步操作

此示例策略將對 Amazon Textract 中同步操作的訪問權限授予AWS帳户。

```
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "textract:DetectDocumentText",
                "textract:AnalyzeDocument"
            ],
            "Resource": "*"
        }
    ]
```

允許訪問 Amazon Textract 中的異步操作

以下示例策略為AWS賬户訪問 Amazon Textract 中使用的所有異步操作。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "textract:StartDocumentTextDetection",
                "textract:StartDocumentAnalysis",
```

```
                "textract:GetDocumentTextDetection",
                "textract:GetDocumentAnalysis"
            ],
            "Resource": "*"
        }
    ]
}
```

針對 Amazon Textract Identity to Access 進行故

請使用以下資訊來協助您診斷和修正使用 Amazon Textract 和 IAM 時可能遇到的常見問題。

主題

- 我未獲授權，不得在 Amazon Textract 中執行動作
- 我未獲得執行 iam：PassRole 的授權
- 我想要檢視我的存取金鑰
- 我是管理員，想要允許其他人存取 Amazon Textract
- 我想要允許外部的人AWS用於訪問我的 Amazon Textract 資源的賬戶

我未獲授權，不得在 Amazon Textract 中執行動作

若 AWS Management Console 告知您並未獲得執行動作的授權，您必須聯絡您的管理員以取得協助。您的管理員是提供您使用者名稱和密碼的人員。

下面的示例錯誤發生在`mateojackson`IAM 用户嘗試運行`DetectDocumentText`在測試圖像上，但沒有`textract:DetectDocumentText`許可。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 textract:DetectDocumentText on resource: textimage.png
```

在此情況下，Mateo 會請求管理員更新他的政策，允許他使用 `textimage.png` 動作存取 `textract:DetectDocumentText` 資源。

我未獲得執行 iam：PassRole 的授權

若您收到錯誤，告知您並未獲得執行 `iam:PassRole` 動作的授權，您必須聯絡您的管理員以取得協助。您的管理員是提供您使用者名稱和密碼的人員。要求該人員更新您的政策，允許您將角色傳遞給 Amazon Textract。

有些 AWS 服務允許您傳遞現有的角色至該服務，而無須建立新的服務角色或服務連結角色。若要執行此作業，您必須擁有將角色傳遞至該服務的許可。

以下範例錯誤會在名為marymajor嘗試使用主控台在 Amazon Textract 中執行動作。但是，動作要求服務具備服務角色授予的許可。Mary 沒有將角色傳遞至該服務的許可。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在這種情況下，Mary 會請求管理員更新她的政策，允許她執行 iam:PassRole 動作。

我想要檢視我的存取金鑰

在您建立 IAM 使用者存取金鑰後，您可以隨時檢視您的存取金鑰 ID。但是，您無法再次檢視您的私密存取金鑰。若您遺失了秘密金鑰，您必須建立新的存取金鑰對。

存取金鑰包含兩個部分：存取金鑰 ID (例如 AKIAIOSFODNN7EXAMPLE) 和私密存取金鑰 (例如 wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY)。如同使用者名稱和密碼，您必須一起使用存取金鑰 ID 和私密存取金鑰來驗證您的請求。就如對您的使用者名稱和密碼一樣，安全地管理您的存取金鑰。

> **Important**
>
> 請勿將您的存取金鑰提供給第三方，甚至是協助尋找您的正式使用者 ID。執行此作業，可能會讓他人能夠永久存取您的帳戶。

建立存取金鑰對時，您會收到提示，要求您將存取金鑰 ID 和私密存取金鑰儲存在安全位置。私密存取金鑰只會在您建立它的時候顯示一次。若您遺失了私密存取金鑰，您必須將新的存取金鑰新增到您的 IAM 使用者。您最多可以擁有兩個存取金鑰。若您已有兩個存取金鑰，您必須先刪除其中一個金鑰對，才能建立新的金鑰對。若要檢視說明，請參閱《IAM 使用者指南》中的管理存取金鑰。

我是管理員，想要允許其他人存取 Amazon Textract

若要允許其他人存取 Amazon Textract，您必須針對需要存取的人員或應用程式建立 IAM 實體 (使用者或角色)。他們將使用該實體的憑證來存取 AWS。您接著必須將政策連接到實體，在 Amazon Textract 中授予正確的許可。

若要立即開始使用，請參閱《IAM 使用者指南》中的建立您的第一個 IAM 委派使用者及群組。

我想要允許外部的人AWS用於訪問我的 Amazon Textract 資源的賬户

您可以建立一個角色，讓其他帳戶中的使用者或您組織外部的人員存取您的資源。您可以指定要允許哪些信任對象取得該角色。針對支援基於資源的政策或存取控制清單 (ACL) 的服務，您可以使用那些政策來授予人員存取您資源的許可。

若要進一步了解，請參閱以下內容：

- 若要了解 Amazon Textract 是否支援這些功能，請參Amazon Textract 如何搭配 IAM 運作。
- 若要了解如何存取您擁有的所有 AWS 帳戶 所提供的資源，請參閱《IAM 使用者指南》中的將存取權提供給您所擁有的另一個 AWS 帳戶 中的 IAM 使用者。
- 若要了解如何將資源的存取權提供給第三方 AWS 帳戶，請參閱《IAM 使用者指南》中的將存取權提供給第三方擁有的 AWS 帳戶。
- 若要了解如何透過聯合身分提供存取權，請參閱《IAM 使用者指南》中的將存取權提供給在外部進行身分驗證的使用者 (聯合身分)。
- 若要了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的 IAM 角色與資源型政策的差異。

記錄和監控

要監控 Amazon Textract，請使用 Amazon CloudWatch。本節提供有關如何設定 Amazon Textract 監控的資訊。它還提供了 Amazon Textract 指標的參考內容。

主題

- 監控 Amazon Textract
- 存取 Amazon Textract 的 CloudWatch 指標

監控 Amazon Textract

您可以透過 CloudWatch，為您的帳户取得個別 Amazon Textract 操作的指標或全域 Amazon Textract 指標。您可以使用這些指標來追蹤 Amazon Textart 解決方案的運作狀態，並設定警示，在一或多個指標落在定義閾值以外時通知您。例如，您可以查看發生的伺服器錯誤次數指標。您也可以查看特定 Amazon Textract 操作的成功次數指標。若要查看指標，您可以使用Amazon CloudWatch，AWS CLI，或CloudWatch API。

使用 CloudWatch 指 Amazon Textract

要使用指標，您必須指定下列資訊：

- 指標維度或者無維度。維度是一組用來單獨辨識指標的名稱值組。Amazon Textract 有一個維度，稱為操作。它提供特定操作的指標。如果您未指定維度，指標範圍將適用於帳號內的所有 Amazon Textract 操作。
- 指標名稱，例如 `UserErrorCount`。

您可以 Amazon Textract 用AWS Management Console，AWS CLI或 CloudWatch API。您也可以透過其中一個 Amazon AWS 軟體開發套件 (SDK) 或 CloudWatch API 工具來使用 CloudWatch API。主控台會根據 CloudWatch API 的原始資料顯示一系列圖形。根據需求，您可能偏好使用顯示於主控台中的圖形或自 API 擷取的圖形。

下列清單顯示一些常見的指標用途。這些是協助您開始的建議，而不是完整清單。

我要如何？	相關指標
我要如何得知我的應用程式已達每秒最高請求數量？	監控 `ThrottledCount` 指標的 `Sum` 數據。
我要如何監控請求錯誤？	使用 `UserErrorCount` 指標的 `Sum` 統計資料。
我要如何找到請求總數？	使用 `ResponseTime` 指標的 `SampleCount` 統計資料。其中包括任何產生錯誤的請求。如果您只想要查看成功操作呼叫，請使用 `SuccessfulRequestCount` 指標。
如何監控 Amazon Textract 操作呼叫的延遲？	使用 `ResponseTime` 指標。

您必須擁有適當的 CloudWatch 許可才可使用 CloudWatch 來監控 Amazon Textract。如需詳細資訊，請參閱 Amazon CloudWatch 身分驗證與存取控制。

存取 Amazon Textract 標

以下範例展示如何使用 CloudWatch 主控台、AWS CLI和 CloudWatch API。

檢視指標 (主控台)

1. 透過 https://console.aws.amazon.com/cloudwatch/ 開啟 CloudWatch 主控台。
2. 選擇指標中，選擇所有指標選項卡，然後選擇Amazon Textract。
3. 選擇按操作，然後選擇一個指標。

   例如，您可以選擇StartDocumentAnalysis度量來衡量已啟動異步文檔分析的次數。
4. 選擇日期範圍的值。指標計數顯示於圖形中。

若要成功查看指標**StartDocumentAnalysis**在一段時間內發出的操作呼叫 (CLI)

- 開啟 AWS CLI，然後輸入下列命令：

```
aws cloudwatch get-metric-statistics \
    --metric-name SuccessfulRequestCount \
    --start-time 2019-02-01T00:00:00Z \
    --period 3600 \
    --end-time 2019-03-01T00:00:00Z \
    --namespace AWS/Textract \
    --dimensions Name=Operation,Value=StartDocumentAnalysis \
    --statistics Sum
```

  此範例顯示在一段時間內執行的成功 `StartDocumentAnalysis` 操作呼叫。如需詳細資訊，請參閱 get-metric-statistics。

若要存取指標 (CloudWatch API)

- 呼叫 `GetMetricStatistics`。如需詳細資訊，請參閱 。Amazon CloudWatch API 參考。

建立警示

您可以建立 CloudWatch 警示，在警示變更狀態時傳送 Simple Notification Service (Amazon SNS) 訊息。警示會在您指定的期間監看單一指標。警示會根據在數個期間與指定閾值相關的指標值，來執行一個或多個動作。此動作是傳送到 Amazon SNS 主題或 Auto Scaling 政策的通知。

警示僅會針對持續狀態變更呼叫動作。CloudWatch 警示不會僅因為它們處於特定狀態而叫用動作。狀態必須發生變更並維持一段指定的時間。

若要設定警示 (主控台)

1. 登入 AWS Management Console 並開啟位於 https://console.aws.amazon.com/cloudwatch/ 的 CloudWatch 主控台。
2. 在導覽窗格中，選擇警報，然後選擇建立警示。此操作可開啟建立警示精靈。
3. 選擇 Select metric (選取指標)。
4. 在 中所有指標選項卡上選擇Textract。
5. 選擇按操作，然後選擇一個指標。

   例如，您可以選擇StartDocumentAnalysis可設定異步文件分析操作的最高次數警示。
6. 選擇 Graphed metrics (圖表化指標) 標籤。
7. 在 Statistic (統計資料) 中選擇 Sum (總和)。
8. 選擇 Select metric (選取指標)。
9. 填入 Name (名稱) 和 Description (說明)。對於 Whenever (每當) 選項，請選擇 >= 並輸入您所選擇的最大值。
10. 如果您希望 CloudWatch 在達到警示狀態時傳送電子郵件給您，請在每當此警報時：，選擇狀態為警報。要向現有的 Amazon SNS 主題發送警報，對於發送通知至：下，選擇現有 SNS 主題。若要設定新的電子郵件訂清單的名稱與電子郵件地址，請選擇新列表。CloudWatch 會儲存清單並顯示於欄位中，可提供未來設定警示時使用。

    > **Note**
    >
    > 如果您使用新列表若要建立新的 Amazon SNS 主題，電子郵件地址必須先經過驗證，目標收件人才可接收通知。Amazon SNS 只會在警示進入警示狀態時才會傳送電子郵件。如果此警示狀態在驗證電子郵件地址之前發生變更，目標收件人就不會收到通知。
11. 選擇 Create Alarm (建立警示)。

設定警示 (AWS CLI)

- 開啟 AWS CLI，然後輸入下列命令。變更的值`alarm-actions`參數，以連結您之前建立的 Amazon SNS 主題。

```
aws cloudwatch put-metric-alarm \
    --alarm-name StartDocumentAnalysisUserErrors \
```

```
    --alarm-description "Alarm when more than 10 StartDocumentAnalysys user errors
occur within 5 minutes" \
    --metric-name UserErrorCount \
    --namespace AWS/Textract \
    --statistic Sum \
    --period 300 \
    --threshold 10 \
    --comparison-operator GreaterThanThreshold \
    --evaluation-periods 1 \
    --unit Count \
    --dimensions Name=Operation,Value=StartDocumentAnalysis \
    --alarm-actions arn:aws:sns:us-east-1:111111111111:alarmtopic
```

此範例說明如何為呼叫在 5 分鐘內發生 10 次以上的用户錯誤時通知`StartDocumentAnalysis`。如需詳細資訊，請參閱 put-metric-alarm。

若要設定警示 (CloudWatch API)

- 呼叫 `PutMetricAlarm`。如需詳細資訊，請參閱「」Amazon CloudWatch API 參考。

存取 Amazon Textract 的 CloudWatch 指標

本節包含 Amazon CloudWatch 指標和操作維度可用於 Amazon Textract。

您也可以從 Amazon Textract 主控台查看 Amazon Textract 指標的彙總檢視。

存取 Amazon Textract 的 CloudWatch 指標

下表顯示 Amazon Textract 指標摘要。

指標	描述
SuccessfulRequestCount	成功請求的數量。成功請求的回應碼範圍是 200 到 299。 單位：計數 有效的統計資訊：`Sum,Average`

指標	描述
ThrottledCount	已調節的請求數目。Amazon Textract 會在收到的請求超過為您的帳户每秒所設的交易上限時節制請求。如果經常超過為您的帳戶所設的限制，您可以請求提高上限。若要請求提高，請參閱 AWS Service Limits。 單位：計數 有效的統計資訊：Sum,Average
ResponseTime	Amazon Textract 計算回應的時間（以毫秒為單位）。 個單位: 1. Data Samples 統計資料的計數 2. Average 統計資料的毫秒 有效的統計資訊：Data Samples,Average **Note** 所以此ResponseTime 指標不包含在 Amazon Textract 指標窗格中。
ServerErrorCount	伺服器錯誤的次數。伺服器錯誤的回應碼範圍是 500 到 599。 單位：計數 有效的統計資訊：Sum,Average
UserErrorCount	使用者錯誤次數 (無效參數、無效影像、無權限等)。使用者錯誤的回應碼範圍是 400 到 499。 單位：計數 有效的統計資訊：Sum,Average

Amazon Textract 的 CloudWatch 維度

若要擷取特定操作的指標，請使用 `AWS/Textract` 命名空間並提供操作維度。如需維度的詳細資訊，請參維度中的Amazon CloudWatch 使用者指南。

使用 記錄 Amazon Textract API 呼叫AWS CloudTrail

Amazon Textract 系統與AWS CloudTrail，該服務可提供記錄使用者、角色或AWS服務在 Amazon Textract。CloudTrail 會將 Amazon Textract 的所有 API 呼叫捕獲為事件。捕獲的呼叫包括從 Amazon Textract 主控台執行的呼叫，以及對 Amazon Textract API 作業發出的程式碼呼叫。

如果您建立線索，您可以持續傳送 CloudTrail 事件至 Amazon S3 儲存貯體，包括 Amazon Textract 的事件。即使您未設定追蹤，依然可以透過 CloudTrail 主控台中的 Event history (事件歷史記錄) 檢視最新事件。您可以利用 CloudTrail 收集到的資訊，判斷已對 Amazon Textract 提出的請求、提出請求的 IP 地址、何人出要求、提出請求的時間，以及其他詳細資訊。

若要進一步了解 CloudTrail，請參閱 AWS CloudTrail 使用者指南。

CloudTrail 中的 Amazon Textract CloudTrail 信息

當您建立帳戶時，系統即會在 AWS 帳戶中啟用 CloudTrail。在 Amazon Textract 中發生活動時，系統便會將該活動記錄至 CloudTrail 事件，並將其他AWS中的服務事件事件歷史記錄。您可以檢視、搜尋和下載 AWS 帳戶的最新事件。如需詳細資訊，請參閱使用 CloudTrail 事件歷史記錄檢視事件。

若要持續記錄AWS帳户（包含 Amazon Textract 的事件），建立線索。追蹤能讓 CloudTrail 將日誌檔交付至 Amazon S3 儲存貯體。根據預設，當您在主控台建立追蹤記錄時，追蹤記錄會套用到所有 AWS 區域。該追蹤會記錄來自 AWS 分割區中所有區域的事件，並將日誌檔案交付到您指定的 Amazon S3 儲存貯體。此外，您可以配置其他AWS服務來進一步分析和處理 CloudTrail 日誌中所收集的事件資料。如需詳細資訊，請參閱下列內容：

- 建立追蹤的概觀
- CloudTrail 支援的服務和整合
- 設定 CloudTrail 的 Amazon SNS 通知
- 從多個區域接收 CloudTrail 日誌檔案，以及從多個帳戶接收 CloudTrail 日誌檔案

所有 Amazon Textract 操作都是由 CloudTrail 記錄並記載於API 參考。例如，對 `DetectDocumentText`、`AnalyzeDocument` 以及 `GetDocumentText` 動作發出的呼叫會在 CloudTrail 日誌檔案中產生項目。

每一筆事件或日誌項目都會包含產生請求者的資訊。身分資訊可協助您判斷下列事項：

- 該請求是否透過根或 AWS Identity and Access Management (IAM) 使用者憑證來提出。
- 提出該要求時，是否使用了特定角色或聯合身分使用者的暫時安全憑證。
- 該請求是否由另一項 AWS 服務提出。

如需詳細資訊，請參閱 CloudTrail userIdentity 元素。

請求參數和未記錄的響應字段

出於隱私目的，某些請求參數和響應字段不會記錄，例如請求圖像字節或響應邊界框信息。請求參數中提供的 Amazon S3 存儲桶名稱和文件名在 CloudTrail 日誌條目中提供。在 CloudTrail 日誌中不提供有關請求中傳遞的圖像字節的信息。下表顯示了每個 Amazon Textract 操作未記錄的輸入參數和響應參數。

操作	請求參數	回應欄位
AnalyzeDocument	`Bytes`	全部
DetectDocumentText	`Bytes`	全部
StartDocumentAnalysis	None (無)	None (無)
GetDocumentAnalysis	None (無)	全部
StartDocumentTextDetection	None (無)	None (無)
GetDocumentTextDetection	None (無)	全部

了解 Amazon Textract 日誌檔案項目

追蹤是一種組態，能讓事件以日誌檔案的形式交付到您指定的 Amazon S3 儲存貯體。CloudTrail 日誌檔包含一或多個日誌項目。一個事件代表任何來源提出的單一請求，並包含所請求之操作的相關資訊、操作的日期和時間、請求參數等等。CloudTrail 日誌檔案並非依公有 API 呼叫的堆疊追蹤排序，因此不會以任何特定順序出現。

以下範例顯示的 CloudTrail 日誌項目會示範 `AnalyzeDocument` 操作：輸入的圖像字節`document`和分析結果（`responseElements`）未記錄。

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::111111111111:user/janedoe",
        "accountId": "111111111111",
        "accessKeyId": "AIDACKCEVSQ6C2EXAMPLE",
        "userName": "janedoe"
    },
    "eventTime": "2019-04-03T23:56:31Z",
    "eventSource": "textract.amazonaws.com",
    "eventName": "AnalyzeDocument",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "198.51.100.0",
    "userAgent": "",
    "requestParameters": {
        "document": {},
        "featureTypes": [
            "TABLES"
        ]
    },
    "responseElements": null,
    "requestID": "e387676b-d1f0-4ea7-85d6-f5a344052dce",
    "eventID": "c5db79ce-e4ea-4401-8517-784481d559f7",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111111111111"
}
```

以下範例顯示StartDocumentAnalysisoperation. 日誌條目包括 Amazon S3 存儲桶名稱和映像文件名稱documentLocation。日誌還包括操作響應。

```
{
    "Records": [
        {
            "eventVersion": "1.05",
            "userIdentity": {
                "type": "IAMUser",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::111111111111:user/janedoe",
                "accountId": "11111111111",
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
                "userName": "janedoe"
```

```
            },
            "eventTime": "2019-04-04T01:42:24Z",
            "eventSource": "textract.amazonaws.com",
            "eventName": "StartDocumentAnalysis",
            "awsRegion": "us-east-1",
            "sourceIPAddress": "198.51.100.0",
            "userAgent": "",
            "requestParameters": {
                "documentLocation": {
                    "s3Object": {
                        "bucket": "bucket",
                        "name": "document.png"
                    }
                },
                "featureTypes": [
                    "TABLES"
                ]
            },
            "responseElements": {
                "jobId":
 "f3c718b444fa603d5d625ab967008f4b620d4650c9db8ca1cae01ef7efe51373"
            },
            "requestID": "9ae352e8-9de1-41ad-b77b-85aa348c2e82",
            "eventID": "f741bca0-c3cb-4805-82ea-baf76439deef",
            "eventType": "AwsApiCall",
            "recipientAccountId": "111111111111"
        }

    ]
}
```

Amazon Textract 的合規驗證

第三方稽核人員評估 Amazon Textract 的安全與合規AWS合規計劃。其中包括 HIPAA、SOC、ISO 和 PCI。

> **Note**
>
> 如果您通過 Textract 服務處理受 PCI DSS 合規性約束的數據，則必須聯繫 AWS Support 並遵循向您提供的流程來選擇退出您的賬戶。

如需特定合規計劃範圍內的 AWS 服務清單，請參閱合規計劃範圍內的 AWS 服務。如需一般資訊，請參閱 AWS 合規計劃。

您可使用 AWS Artifact 下載第三方稽核報告。如需詳細資訊，請參閱下載 AWS Artifact 中的報告。

您使用 Amazon Textract 的合規責任，取決於資料的機密性、您公司的合規目標及適用法律和法規。AWS提供下列資源，以協助合規：

- 安全與合規快速入門指南 – 這些部署指南討論架構考量，並提供在 AWS 上部署以安全及合規為重心之基準環境的步驟。
- HIPAA 安全與合規架構白皮書 – 本白皮書說明公司可如何運用 AWS 來建立 HIPAA 合規的應用程式。
- AWS 合規資源 – 這組手冊和指南可能適用於您的產業和位置。
- AWS Config 開發人員指南中的使用規則評估資源：AWS Config 服務可評估資源組態對於內部實務、業界準則和法規的合規狀態。
- AWS Security Hub— 這AWS服務可供您全面檢視中的安全狀態AWS。此安全中心可助您檢查是否符合安全產業標準和最佳實務。

Amazon Textract 的恢復能力

AWS 全球基礎設施是以 AWS 區域與可用區域為中心建置的。AWS區域提供多個分開且隔離的實際可用區域，並以低延遲、高輸送量和高度備援聯網功能相互連結。透過可用區域，您可以設計與操作的應用程式和資料庫，在可用區域之間自動容錯移轉而不會發生中斷。可用區域的可用性、容錯能力和擴充能力，均較單一或多個資料中心的傳統基礎設施還高。

如需有關 AWS 區域與可用區域的詳細資訊，請參閱 AWS 全域基礎設施。

> **Note**
>
> 根據《通用數據保護條例》(GDPR)，不允許跨區域傳輸數據。

Amazon Textract 的基礎設施安全

作為一種受管服務，Amazon Textract 受AWS全局網絡安全過程，詳情請參閱Amazon Web Services：安全流程概觀白皮書。

您使用AWS發佈的 API 呼叫，透過網路存取 Amazon Textract。用戶端必須支援 Transport Layer Security (TLS) 1.0 或更新版本。建議使用 TLS 1.2 或更新版本。用戶端也必須支援具備完美轉送私密 (PFS) 的密碼套件，例如臨時 Diffie-Hellman (DHE) 或橢圓曲線臨時 Diffie-Hellman (ECDHE)。現代系統 (如 Java 7 和更新版本) 大多會支援這些模式。

此外，請求必須使用存取金鑰 ID 和與 IAM 委託人相關聯的私密存取金鑰來簽署。或者，您可以使用 AWS Security Token Service (AWS STS) 來產生暫時安全憑證來簽署請求。

Amazon Textract 中的組態與漏洞分析

組態和 IT 控制是 AWS 與身為我們客戶的您共同的責任。如需詳細資訊，請參閱 AWS 共同的責任模型。

Amazon Textract 和界面 VPC 端點 (AWS PrivateLink)

您可以在 VPC 與 Amazon Textract 之間建立私有連線，方法是建立界面 VPC 端點。界面端點採用的技術AWS PrivateLink，這項技術可讓您以私有方式存取 Amazon Textract API，無需透過網際網路閘道、NAT 裝置、VPN 連線或 AWS Direct Connect 連線。您 VPC 中的實例不需要公有 IP 地址，即能與 Amazon Textract API 通訊。您的 VPC 與 Amazon Textract 之間的網路流量都會在 AWS 網路的範圍內。

每個介面端點都是由您子網路中的一或多個彈性網路介面表示。

如需詳細資訊，請參閱「」介面 VPC 端點 (AWS PrivateLink)中的Amazon VPC User Guide。

Amazon Textract VPC 端點的考量事項

在設定 Amazon Textract 的界面 VPC 端點前，請務必檢界面端點屬性和限制中的Amazon VPC User Guide。

Amazon Textract 支援從您的 VPC 呼叫其所有 API 動作。

為 Amazon Textract 建立界面 VPC 端點

您可使 Amazon Textract VPC 主控台或AWS Command Line Interface(AWS CLI。如需詳細資訊，請參閱 Amazon VPC 使用者指南中的建立介面端點。

使用下列服務名稱為 Amazon Textract 建立 VPC 端點：

- 通過亞馬遜。##.textract-用於為大多數 Amazon Textract 操作創建終端節點。

- 通過亞馬遜。##.textract-fips-為 Amazon Textract 建立符合聯邦資訊處理標準 (FIPS) 第 140-2 號公報美國政府標準的端點。

如果您對該端點啟用私有 DNS，您可以使用其區域的預設 DNS 名稱向 Amazon Textract 發出 API 請求，例如`textract.us-east-1.amazonaws.com`。

如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 透過介面端點存取服務。

為 Amazon Textract 建立 VPC 端點政策

您可以將端點政策連接至控制 Amazon Textract 存取權的 VPC 端點。此政策會指定下列資訊：

- 可執行動作的委託人。
- 可執行的動作。
- 可供執行動作的資源。

如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的使用 VPC 端點控制對服務的存取。

範例：Amazon Textract 動作的 VPC 端點政策

以下是 Amazon Textract 端點政策的範例。連接至端點後，此政策會針對所有資源上的所有委託人，授予列出的 Amazon Textract 動作的存取權限。

此範例政策只會允許存取操作`DetectDocumentText`和`AnalyzeDocument`。使用者仍然可以在 VPC 端點之外呼叫 Amazon Textract 操作。

```
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "textract:DetectDocumentText",
            "textract:AnalyzeDocument",
         ],
         "Resource":"*"
      }
   ]
}
```

API 參考

本節提供 Amazon Textract API 操作的文件。

主題

動作

支援以下動作：

- AnalyzeDocument
- AnalyzeExpense
- AnalyzeID
- DetectDocumentText
- GetDocumentAnalysis
- GetDocumentTextDetection
- GetExpenseAnalysis
- StartDocumentAnalysis
- StartDocumentTextDetection
- StartExpenseAnalysis

AnalyzeDocument

分析輸入檔案，分析已偵測項目之間的關係。

返回的信息類型如下：

- 資料表單（金鑰值對）。有關資訊會以兩種方式傳回Block對象, 每個類型KEY_VALUE_SET：金鑰Block對象和一個值Block物件。例如：名稱: 安娜·席爾瓦·卡羅萊納州包含一個鍵和值。名稱:是關鍵。安娜·席爾瓦·卡羅萊納州均為值。
- 表格和表格單元格數據。資料表Block對象包含有關檢測到的表的信息。儲存格Block對象返回表中的每個單元格。
- 文本的行和單詞。行Block物件包含一個或多個 WORDBlock物件。返回文檔中檢測到的所有行和單詞（包括與FeatureTypes。

選擇元素（如複選框和選項按鈕（單選按鈕）可以在表單數據和表格中檢測。選擇元素Block對象包含有關選擇元素的信息，包括選擇狀態。

您可以選擇要執行的分析類型，方法是指定FeatureTypes列表。

輸出返回在Block物件。

AnalyzeDocument均為同步操作。要異步分析文檔，請使用StartDocumentAnalysis。

如需詳細資訊，請參閱「」文件文字分析。

請求語法

```
{
   "Document": {
      "Bytes": blob,
      "S3Object": {
         "Bucket": "string",
         "Name": "string",
         "Version": "string"
      }
   },
   "FeatureTypes": [ "string" ],
   "HumanLoopConfig": {
      "DataAttributes": {
         "ContentClassifiers": [ "string" ]
      },
```

```
        "FlowDefinitionArn": "string",
        "HumanLoopName": "string"
    }
}
```

請求參數

請求接受採用 JSON 格式的下列資料。

Document

輸入文檔為 base64 編碼的字節或 Amazon S3 對象。如果您使用 AWS CLI 調用 Amazon Textract 操作，則無法傳遞圖像字節。文檔必須是 JPEG、PNG、PDF 或 TIFF 格式的圖像。

如果您使用 AWS 開發工具包調用 Amazon Textract，則可能不需要對使用`Bytes`欄位。

類型：Document 物件

：必要 是

FeatureTypes

要執行的分析類型的列表。將 TABLE 添加到列表以返回有關輸入文檔中檢測到的表的信息。添加表單以返回檢測到的表單數據。要執行這兩種類型的分析，請將表和表格添加到`FeatureTypes`。文檔中檢測到的所有行和單詞都包含在響應中（包括與`FeatureTypes`。

類型：字串陣列

有效值： `TABLES | FORMS`

：必要 是

HumanLoopConfig

設置用於分析文檔的循環工作流的配置。

類型：HumanLoopConfig 物件

：必要 否

回應語法

```
{
   "AnalyzeDocumentModelVersion": "string",
   "Blocks": [
```

```
      {
         "BlockType": "string",
         "ColumnIndex": number,
         "ColumnSpan": number,
         "Confidence": number,
         "EntityTypes": [ "string" ],
         "Geometry": {
            "BoundingBox": {
               "Height": number,
               "Left": number,
               "Top": number,
               "Width": number
            },
            "Polygon": [
               {
                  "X": number,
                  "Y": number
               }
            ]
         },
         "Id": "string",
         "Page": number,
         "Relationships": [
            {
               "Ids": [ "string" ],
               "Type": "string"
            }
         ],
         "RowIndex": number,
         "RowSpan": number,
         "SelectionStatus": "string",
         "Text": "string",
         "TextType": "string"
      }
   ],
   "DocumentMetadata": {
      "Pages": number
   },
   "HumanLoopActivationOutput": {
      "HumanLoopActivationConditionsEvaluationResults": "string",
      "HumanLoopActivationReasons": [ "string" ],
      "HumanLoopArn": "string"
   }
```

```
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

AnalyzeDocumentModelVersion

用於分析文件的模型版本。

類型：字串

Blocks

檢測和分析的項目`AnalyzeDocument`。

類型：的陣列Block對象

DocumentMetadata

有關已分析文檔的元數據。一個例子是頁數。

類型：DocumentMetadata 物件

HumanLoopActivationOutput

顯示循環評估中人類的結果。

類型：HumanLoopActivationOutput 物件

錯誤

AccessDeniedException

您未獲授權執行動作。使用已獲授權之使用者或 IAM 角色的 Amazon Resource Name (ARN) 來執行操作。

HTTP 狀態碼：400

BadDocumentException

Amazon Textract 無法閱讀該文檔。有關 Amazon Textract 中文檔限制的更多信息，請參閱亞馬遜文字中的硬性限制。

HTTP 狀態碼：400

DocumentTooLargeException

無法處理該文檔，因為它太大。用於同步操作的上限文件大小，10 MB。對於 PDF 文件，異步操作的最大文檔大小為 500 MB。

HTTP 狀態碼：400

HumanLoopQuotaExceededException

指示您已超過可用循環工作流中活動人工的上限數量

HTTP 狀態碼：400

InternalServerError

Amazon Textract 發生服務問題。請再次嘗試呼叫。

HTTP 狀態碼：500

InvalidParameterException

輸入參數違反限制。例如，在同步操作中，`InvalidParameterException`異常發生時，`S3Object`或者`Bytes`值提供在`Document`請求參數。請驗證您的參數，然後再次呼叫 API 操作。

HTTP 狀態碼：400

InvalidS3ObjectException

Amazon Textract 無法存取請求中指定的 S3 物件。有關詳細信息，請配置對 Amazon S3 的存取如需故障診斷資訊，請參閱 。故障診斷 Amazon S3

HTTP 狀態碼：400

ProvisionedThroughputExceededException

請求數超過您的傳輸量限制。如果您希望提高此限制，請聯絡 Amazon Textract。

HTTP 狀態碼：400

ThrottlingException

Amazon Textract 暫時無法處理請求。請再次嘗試呼叫。

HTTP 狀態碼：500

UnsupportedDocumentException

不支持輸入檔案的格式。操作文檔可以採用 PNG、JPEG、PDF 或 TIFF 格式。

HTTP 狀態碼：400

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- AWS Command Line Interface
- 適用於 .NET 的 AWS 開發套件
- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- AWS適用於 Java 的開發套件第 2 版
- 適用於 JavaScript 的 AWS 開發套件
- 適用於 PHP V3 的 AWS 開發套件
- 適用於 Python 的 AWS 開發套件
- 適用於 Ruby V3 的 AWS 開發套件

AnalyzeExpense

AnalyzeExpense同步分析輸入文檔之間的財務相關關係。

信息返回為ExpenseDocuments並分開如下。

- LineItemGroups-包含LineItems存儲有關文本行的信息，例如購買的物料及其收據上的價格。
- SummaryFields-包含收據的所有其他信息，例如題頭信息或供應商名稱。

請求語法

```
{
   "Document": {
      "Bytes": blob,
      "S3Object": {
         "Bucket": "string",
         "Name": "string",
         "Version": "string"
      }
   }
}
```

請求參數

請求接受採用 JSON 格式的下列資料。

Document

輸入文檔，作為字節或 S3 對象。

您可以使用Bytes屬性。例如，您需要使用Bytes屬性傳遞從本地文件系統加載的文檔。通過使用Bytes屬性必須經過 base64 編碼。如果您使用 AWS 開發工具包調用 Amazon Textract API 操作，則您的代碼可能無需對文檔文件字節進行編碼。

您可以使用 S3 儲存貯體中的影像來傳遞至 Amazon Textract API 操作，方法是使用S3Object屬性。存放在 S3 儲存貯體中的檔案，不需要 base64 編碼。

含有 S3 物件的 S3 儲存貯體區域必須符合您用於 Amazon Textract 操作的 AWS 區域。

如果您使用 AWS CLI 呼叫 Amazon Textract 操作，則不支援使用 Bytes 屬性來傳遞影像位元組。您必須先將文件上傳至 Amazon S3 儲存貯體，再使用 S3Object 屬性呼叫操作。

若要 Amazon Textract 處理 S3 物件，用戶必須具有 S3 物件的存取許可。

類型：Document 物件

：必要 是

回應語法

```
{
   "DocumentMetadata": {
      "Pages": number
   },
   "ExpenseDocuments": [
      {
         "ExpenseIndex": number,
         "LineItemGroups": [
            {
               "LineItemGroupIndex": number,
               "LineItems": [
                  {
                     "LineItemExpenseFields": [
                        {
                           "LabelDetection": {
                              "Confidence": number,
                              "Geometry": {
                                 "BoundingBox": {
                                    "Height": number,
                                    "Left": number,
                                    "Top": number,
                                    "Width": number
                                 },
                                 "Polygon": [
                                    {
                                       "X": number,
                                       "Y": number
                                    }
                                 ]
                              },
                              "Text": "string"
                           },
                           "PageNumber": number,
                           "Type": {
                              "Confidence": number,
```

```
                        "Text": "string"
                     },
                     "ValueDetection": {
                        "Confidence": number,
                        "Geometry": {
                           "BoundingBox": {
                              "Height": number,
                              "Left": number,
                              "Top": number,
                              "Width": number
                           },
                           "Polygon": [
                              {
                                 "X": number,
                                 "Y": number
                              }
                           ]
                        },
                        "Text": "string"
                     }
                  }
               ]
            }
         ]
      }
   ],
   "SummaryFields": [
      {
         "LabelDetection": {
            "Confidence": number,
            "Geometry": {
               "BoundingBox": {
                  "Height": number,
                  "Left": number,
                  "Top": number,
                  "Width": number
               },
               "Polygon": [
                  {
                     "X": number,
                     "Y": number
                  }
               ]
            },
```

```
                  "Text": "string"
               },
               "PageNumber": number,
               "Type": {
                  "Confidence": number,
                  "Text": "string"
               },
               "ValueDetection": {
                  "Confidence": number,
                  "Geometry": {
                     "BoundingBox": {
                        "Height": number,
                        "Left": number,
                        "Top": number,
                        "Width": number
                     },
                     "Polygon": [
                        {
                           "X": number,
                           "Y": number
                        }
                     ]
                  },
                  "Text": "string"
               }
            }
         ]
      }
   ]
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

[DocumentMetadata](#)

有關輸入文檔的信息。

類型：[DocumentMetadata](#) 物件

ExpenseDocuments

Amazon Textract 檢測到的費用。

類型：的陣列ExpenseDocument對象

錯誤

AccessDeniedException

您未獲授權執行動作。使用已獲授權之使用者或 IAM 角色的 Amazon Resource Name (ARN) 來執行操作。

HTTP 狀態碼：HTTP 狀態碼：400

BadDocumentException

Amazon Textract 無法閱讀該文檔。有關 Amazon Textract 中文檔限制的更多信息，請參閱亞馬遜文字中的硬性限制。

HTTP 狀態碼：HTTP 狀態碼：400

DocumentTooLargeException

無法處理該文檔，因為它太大。同步操作的最大檔案大小為 10 MB。對於 PDF 文件，異步操作的最大文檔大小為 500 MB。

HTTP 狀態碼：HTTP 狀態碼：400

InternalServerError

Amazon Textract 發生服務問題。請再次嘗試呼叫。

HTTP 狀態碼：HTTP 狀態碼：500

InvalidParameterException

輸入參數違反限制。例如，在同步操作中，`InvalidParameterException`異常發生時，`S3Object`或者`Bytes`值提供在`Document`請求參數。請驗證您的參數，然後再次呼叫 API 操作。

HTTP 狀態碼：HTTP 狀態碼：400

InvalidS3ObjectException

Amazon Textract 無法存取請求中指定的 S3 物件。有關詳細信息，請設定對 Amazon S3 的存取如需故障診斷資訊，請參閱 。故障診斷 Amazon S3

HTTP 狀態碼：HTTP 狀態碼：400

ProvisionedThroughputExceededException

請求數超過您的傳輸量限制。如果您希望提高此限制，請聯絡 Amazon Textract。

HTTP 狀態碼：HTTP 狀態碼：400

ThrottlingException

Amazon Textract 暫時無法處理請求。請再次嘗試呼叫。

HTTP 狀態碼：HTTP 狀態碼：500

UnsupportedDocumentException

不支援輸入檔案的格式。操作文檔可以採用 PNG、JPEG、PDF 或 TIFF 格式。

HTTP 狀態碼：HTTP 狀態碼：400

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- AWS Command Line Interface
- 適用於 .NET 的 AWS 開發套件
- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- AWS適用於 Java 的開發套件第 2 版
- 適用於 JavaScript 的 AWS 開發套件
- 適用於 PHP V3 的 AWS 開發套件
- 適用於 Python 的 AWS 開發套件
- 適用於 Ruby V3 的 AWS 開發套件

AnalyzeID

分析身份證明文件以獲取相關信息。此信息被提取並返回為IdentityDocumentFields，它記錄提取文本的標準化字段和值。與其他 Amazon Textract 操作不同，AnalyzeID不會傳回任何幾何數據。

請求語法

```
{
   "DocumentPages": [
      {
         "Bytes": blob,
         "S3Object": {
            "Bucket": "string",
            "Name": "string",
            "Version": "string"
         }
      }
   ]
}
```

請求參數

請求接受採用 JSON 格式的下列資料。

DocumentPages

傳遞給分析 ID 的文檔。

類型：的陣列Document對象

陣列成員：項目數下限為 1。項目數上限為 2。

的必要項目：是

回應語法

```
{
   "AnalyzeIDModelVersion": "string",
   "DocumentMetadata": {
      "Pages": number
   },
```

```
   "IdentityDocuments": [
      {
         "DocumentIndex": number,
         "IdentityDocumentFields": [
            {
               "Type": {
                  "Confidence": number,
                  "NormalizedValue": {
                     "Value": "string",
                     "ValueType": "string"
                  },
                  "Text": "string"
               },
               "ValueDetection": {
                  "Confidence": number,
                  "NormalizedValue": {
                     "Value": "string",
                     "ValueType": "string"
                  },
                  "Text": "string"
               }
            }
         ]
      }
   ]
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

AnalyzeIDModelVersion

用於處理文檔的分析身份 API 版本。

類型：字串

DocumentMetadata

有關輸入文檔的信息。

類型：DocumentMetadata 物件

IdentityDocuments

分析編號處理的文件列表。包括一個數字，表示它們在列表中的位置和文檔的響應結構。

類型：的陣列IdentityDocument對象

錯誤

AccessDeniedException

您未獲授權執行動作。使用已獲授權之使用者或 IAM 角色的 Amazon Resource Name (ARN) 來執行操作。

HTTP 狀態碼：400

BadDocumentException

Amazon Textract 無法閱讀該文檔。有關 Amazon Textract 中文檔限制的更多信息，請參閱亞馬遜文字中的硬性限制。

HTTP 狀態碼：400

DocumentTooLargeException

無法處理該文檔，因為它太大。同步操作的最大文件大小為 10 MB。對於 PDF 文件，異步操作的最大文檔大小為 500 MB。

HTTP 狀態碼：400

InternalServerError

Amazon Textract 發生服務問題。請再次嘗試呼叫。

HTTP 狀態碼：500

InvalidParameterException

輸入參數違反限制。例如，在同步操作中，`InvalidParameterException`異常發生時，`S3Object`或者`Bytes`值提供在`Document`請求參數。請驗證您的參數，然後再次呼叫 API 操作。

HTTP 狀態碼：400

InvalidS3ObjectException

Amazon Textract 無法存取請求中指定的 S3 物件。有關詳細信息，請設定對 Amazon S3 的存取如需故障診斷資訊，請參閱 。故障診斷 Amazon S3

HTTP 狀態碼：400

ProvisionedThroughputExceededException

請求數超過您的傳輸量限制。如果您希望提高此限制，請聯絡 Amazon Textract。

HTTP 狀態碼：400

ThrottlingException

Amazon Textract 暫時無法處理請求。請再次嘗試呼叫。

HTTP 狀態碼：500

UnsupportedDocumentException

不支持輸入檔案的格式。操作文檔可以採用 PNG、JPEG、PDF 或 TIFF 格式。

HTTP 狀態碼：400

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- AWS Command Line Interface
- 適用於 .NET 的 AWS 開發套件
- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- AWS適用於 Java 的開發套件第 2 版
- 適用於 JavaScript 的 AWS 開發套件
- 適用於 PHP V3 的 AWS 開發套件
- 適用於 Python 的 AWS 開發套件
- 適用於 Ruby V3 的 AWS 開發套件

DetectDocumentText

檢測輸入文檔中的文本。Amazon Textract 可以檢測文本行和構成一行文本的單詞。輸入文檔必須是 JPEG、PNG、PDF 或 TIFF 格式的圖像。DetectDocumentText返回一個數組中檢測到的文本Block物件。

每個文檔頁面都有一個關聯的Block類型為頁面。每頁BlockLINE 的父物件Block對象，表示頁面上檢測到的文本行。一條線Block對象是構成該行的每個單詞的父項。單詞表示為Block類型為 WORD 的對象。

DetectDocumentText是同步操作。要異步分析文檔，請使用StartDocumentTextDetection。

如需詳細資訊，請參閱「」文字偵測。

請求語法

```
{
   "Document": {
      "Bytes": blob,
      "S3Object": {
         "Bucket": "string",
         "Name": "string",
         "Version": "string"
      }
   }
}
```

請求參數

請求接受採用 JSON 格式的下列資料。

Document

輸入文檔為 base64 編碼的字節或 Amazon S3 對象。如果您使用 AWS CLI 調用 Amazon Textract 操作，則無法傳遞圖像字節。文檔必須是 JPEG 或 PNG 格式的圖像。

如果您使用 AWS 開發工具包調用 Amazon Textract，則可能不需要對使用Bytes欄位。

類型：Document 物件

：必要 是

回應語法

```
{
   "Blocks": [
      {
         "BlockType": "string",
         "ColumnIndex": number,
         "ColumnSpan": number,
         "Confidence": number,
         "EntityTypes": [ "string" ],
         "Geometry": {
            "BoundingBox": {
               "Height": number,
               "Left": number,
               "Top": number,
               "Width": number
            },
            "Polygon": [
               {
                  "X": number,
                  "Y": number
               }
            ]
         },
         "Id": "string",
         "Page": number,
         "Relationships": [
            {
               "Ids": [ "string" ],
               "Type": "string"
            }
         ],
         "RowIndex": number,
         "RowSpan": number,
         "SelectionStatus": "string",
         "Text": "string",
         "TextType": "string"
      }
   ],
   "DetectDocumentTextModelVersion": "string",
   "DocumentMetadata": {
      "Pages": number
   }
```

```
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

[Blocks](#)

陣列Block對象，其中包含在文檔中檢測到的文本。

類型：的陣列Block對象

[DetectDocumentTextModelVersion](#)

類型：字串

[DocumentMetadata](#)

有關文檔的元數據。它包含在文檔中檢測到的頁數。

類型：DocumentMetadata 物件

錯誤

AccessDeniedException

您未獲授權執行動作。使用已獲授權之使用者或 IAM 角色的 Amazon Resource Name (ARN) 來執行操作。

HTTP 狀態碼：：400

BadDocumentException

Amazon Textract 無法閱讀該文檔。有關 Amazon Textract 中文檔限制的更多信息，請參閱亞馬遜文字中的硬性限制。

HTTP 狀態碼：：400

DocumentTooLargeException

無法處理該文檔，因為它太大。同步操作的文件大小上限為 10 MB。對於 PDF 文件，異步操作的最大文檔大小為 500 MB。

HTTP 狀態碼：：400

InternalServerError

Amazon Textract 發生服務問題。請再次嘗試呼叫。

HTTP 狀態碼：：500

InvalidParameterException

輸入參數違反限制。例如，在同步操作中，`InvalidParameterException`異常發生時，`S3Object`或者`Bytes`值提供在`Document`請求參數。請驗證您的參數，然後再次呼叫 API 操作。

HTTP 狀態碼：：400

InvalidS3ObjectException

Amazon Textract 無法存取請求中指定的 S3 物件。有關詳細信息，請配置對 Amazon S3 的存取如需故障診斷資訊，請參閱 。故障診斷 Amazon S3

HTTP 狀態碼：：400

ProvisionedThroughputExceededException

請求數超過您的傳輸量限制。如果您希望提高此限制，請聯絡 Amazon Textract。

HTTP 狀態碼：：400

ThrottlingException

Amazon Textract 暫時無法處理請求。請再次嘗試呼叫。

HTTP 狀態碼：：500

UnsupportedDocumentException

不支持輸入文件的格式。操作文檔可以採用 PNG、JPEG、PDF 或 TIFF 格式。

HTTP 狀態碼：：400

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- AWS Command Line Interface

- 適用於 .NET 的 AWS 開發套件
- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- AWS適用於 Java 的開發套件第 2 版
- 適用於 JavaScript 的 AWS 開發套件
- 適用於 PHP V3 的 AWS 開發套件
- 適用於 Python 的 AWS 開發套件
- 適用於 Ruby V3 的 AWS 開發套件

GetDocumentAnalysis

獲取分析文檔中文本的 Amazon Textract 異步操作的結果。

您可以通過調用StartDocumentAnalysis，它返回一個作業標識符（JobId。當文本分析操作完成後，Amazon Textract 會向亞馬遜 Simple Notification Service (Amazon SNS) 主題發佈完成狀態，該主題在首次調用StartDocumentAnalysis。要獲取文本檢測操作的結果，請首先檢查發佈到 Amazon SNS 主題的狀態值是否為SUCCEEDED。如果是這樣，請調用GetDocumentAnalysis，並傳遞作業標識符（JobId）從初始調用到StartDocumentAnalysis。

GetDocumentAnalysis傳回Block物件。傳回下列類型的資訊：

- 資料表單（金鑰值對）。相關資訊會以兩種形式傳回Block對象, 每個類型KEY_VALUE_SET：一個金鑰Block對象和一個值Block物件。例如：名稱: 安娜·席爾瓦·卡羅萊納州包含一個鍵和值。名稱:是關鍵。安娜·席爾瓦·卡羅萊納州是值。
- 表格和表格單元格數據。資料表Block對象包含有關檢測到的表的信息。儲存格Block對象返回表中的每個單元格。
- 文本的行和單詞。一條線Block物件包含一個或多個 WORDBlock物件。返回文檔中檢測到的所有行和單詞（包括與StartDocumentAnalysis FeatureTypes輸入參數）。

選擇元素（如複選框和選項按鈕（單選按鈕）可以在表單數據和表格中檢測。選擇元素Block對象包含有關選擇元素的信息，包括選擇狀態。

使用MaxResults參數來限制傳回的區塊數量。如果結果多於MaxResults，值NextToken在操作響應中包含一個用於取得下一組結果的字符。若要取得下一頁的結果，請調用GetDocumentAnalysis，然後填入NextToken請求參數與上一次調用返回的令牌值GetDocumentAnalysis。

如需詳細資訊，請參閱「」文件文字分析。

請求語法

```
{
   "JobId": "string",
   "MaxResults": number,
   "NextToken": "string"
}
```

請求參數

請求接受採用 JSON 格式的下列資料。

JobId

文字檢測任務的唯一識別符。所以此JobId從StartDocumentAnalysis。一個JobId值僅在 7 天內有效。

類型：字串

長度約束：長度下限為 1。長度上限為 64。

模式：^[a-zA-Z0-9-_]+$

必要：是

MaxResults

每次分頁呼叫可回傳結果的數量上限。您可以指定的最大值為 1,000。如果您指定的值大於 1,000，最多只能傳回 1,000 個結果。預設值為 1,000。

類型：整數

有效範圍：最小值為 1。

必要：否

NextToken

如果之前的響應不完整（因為有更多要檢索的塊），Amazon Textract 會在響應中返回一個分頁令牌。您可以使用此分頁符來檢索下一組區塊。

類型：字串

長度約束：長度下限為 1。長度上限為 255。

模式：.*\S.*

必要：否

回應語法

```
{
   "AnalyzeDocumentModelVersion": "string",
```

```
   "Blocks": [
      {
         "BlockType": "string",
         "ColumnIndex": number,
         "ColumnSpan": number,
         "Confidence": number,
         "EntityTypes": [ "string" ],
         "Geometry": {
            "BoundingBox": {
               "Height": number,
               "Left": number,
               "Top": number,
               "Width": number
            },
            "Polygon": [
               {
                  "X": number,
                  "Y": number
               }
            ]
         },
         "Id": "string",
         "Page": number,
         "Relationships": [
            {
               "Ids": [ "string" ],
               "Type": "string"
            }
         ],
         "RowIndex": number,
         "RowSpan": number,
         "SelectionStatus": "string",
         "Text": "string",
         "TextType": "string"
      }
   ],
   "DocumentMetadata": {
      "Pages": number
   },
   "JobStatus": "string",
   "NextToken": "string",
   "StatusMessage": "string",
   "Warnings": [
      {
```

```
            "ErrorCode": "string",
            "Pages": [ number ]
        }
    ]
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

AnalyzeDocumentModelVersion

類型：字串

Blocks

文本分析操作的結果。

類型：陣列Block對象

DocumentMetadata

有關 Amazon Textract 處理的文檔的信息。DocumentMetadata會在 Amazon Textract 視頻操作的分頁響應的每一頁中返回。

類型：DocumentMetadata 物件

JobStatus

文字檢測任務的目前狀態。

類型：字串

有效值： IN_PROGRESS | SUCCEEDED | FAILED | PARTIAL_SUCCESS

NextToken

如果響應被截斷，Amazon Textract 將返回此令牌。您可以在後續請求中使用此符記來檢索下一組文字檢測結果。

類型：字串

長度約束：長度下限為 1。長度上限為 255。

模式：`.*\S.*`

StatusMessage

如果無法完成檢測任務，則會傳回。包含有關發生什麼錯誤的說明。

類型：字串

Warnings

文檔分析操作期間發生的警告列表。

類型：陣列Warning對象

錯誤

AccessDeniedException

您未獲授權執行動作。使用已獲授權之使用者或 IAM 角色的 Amazon Resource Name (ARN) 來執行操作。

HTTP 狀態碼：400

InternalServerError

Amazon Textract 發生服務問題。請再次嘗試呼叫。

HTTP 狀態碼：500

InvalidJobIdException

將無效的作業標識符傳遞給GetDocumentAnalysis或GetDocumentAnalysis。

HTTP 狀態碼：400

InvalidKMSKeyException

表示您沒有使用輸入的 KMS 密鑰進行解密權限，或者 KMS 密鑰輸入錯誤。

HTTP 狀態碼：400

InvalidParameterException

輸入參數違反限制。例如，在同步操作中，`InvalidParameterException`異常發生時，`S3Object`或者`Bytes`值提供在`Document`請求參數。請驗證您的參數，然後再次呼叫 API 操作。

HTTP 狀態碼：400

InvalidS3ObjectException

Amazon Textract 無法存取請求中指定的 S3 物件。有關詳細信息，請設定對 Amazon S3 的存取如需故障診斷資訊，請參閱 。故障診斷 Amazon S3

HTTP 狀態碼：400

ProvisionedThroughputExceededException

請求數超過您的傳輸量限制。如果您希望提高此限制，請聯絡 Amazon Textract。

HTTP 狀態碼：400

ThrottlingException

Amazon Textract 暫時無法處理請求。請再次嘗試呼叫。

HTTP 狀態碼：500

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- AWS Command Line Interface
- 適用於 .NET 的 AWS 開發套件
- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- AWS適用於 Java 的開發套件第 2 版
- 適用於 JavaScript 的 AWS 開發套件
- 適用於 PHP V3 的 AWS 開發套件
- 適用於 Python 的 AWS 開發套件
- 適用於 Ruby V3 的 AWS 開發套件

GetDocumentTextDetection

獲取用於檢測文檔中文本的 Amazon Textract 異步操作的結果。Amazon Textract 可以檢測文本行和構成一行文本的單詞。

您可以通過調用StartDocumentTextDetection，它返回一個作業標識符（JobId。當文本檢測操作完成後，Amazon Textract 會向亞馬遜 Simple Notification Service (Amazon SNS) 主題發佈完成狀態，該主題在初始調用StartDocumentTextDetection。要獲取文本檢測操作的結果，請首先檢查發佈到 Amazon SNS 主題的狀態值是否為SUCCEEDED。如果是這樣，請調用GetDocumentTextDetection，並傳遞作業標識符（JobId）從初始調用到StartDocumentTextDetection。

GetDocumentTextDetection傳回一個Block物件。

每個文檔頁面都有一個關聯的Block類型為頁面。每頁Block物件是 LINE 的父項Block對象，表示頁面上檢測到的文本行。一條線Block對象是構成該行的每個單詞的父項。單詞表示為Block類型為 WORD 的對象。

使用 MaxResults 參數來限制傳回的塊數量。如果結果多於MaxResults，值NextToken在操作響應中包含一個用於取得下一組結果的分頁符記。若要取得下一頁的結果，請調用GetDocumentTextDetection，然後填入NextToken請求參數與上一次調用返回的令牌值GetDocumentTextDetection。

如需詳細資訊，請參閱「」文件文字偵測。

請求語法

```
{
   "JobId": "string",
   "MaxResults": number,
   "NextToken": "string"
}
```

請求參數

請求接受採用 JSON 格式的下列資料。

JobId

文字偵測任務的唯一識別符。所以此JobId從傳回StartDocumentTextDetection。一個JobId值僅在 7 天內有效。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：^[a-zA-Z0-9-_]+$

：必要 是

MaxResults

每筆分頁呼叫傳回結果的數量上限。您可以指定的最大值為 1,000。如果您指定的值大於 1,000，最多只能傳回 1,000 個結果。預設值為 1,000。

類型：整數

有效範圍：最小值為 1。

：必要 否

NextToken

如果之前的響應不完整（因為有更多要檢索的塊），Amazon Textract 會在響應中返回一個分頁令牌。您可以使用此分頁符記來取回下一組塊。

類型：字串

長度限制：長度下限為 1。長度上限為 255。

模式：.*\S.*

：必要 否

回應語法

```
{
   "Blocks": [
      {
         "BlockType": "string",
         "ColumnIndex": number,
         "ColumnSpan": number,
         "Confidence": number,
         "EntityTypes": [ "string" ],
         "Geometry": {
            "BoundingBox": {
               "Height": number,
```

```
                "Left": number,
                "Top": number,
                "Width": number
             },
             "Polygon": [
                {
                   "X": number,
                   "Y": number
                }
             ]
          },
          "Id": "string",
          "Page": number,
          "Relationships": [
             {
                "Ids": [ "string" ],
                "Type": "string"
             }
          ],
          "RowIndex": number,
          "RowSpan": number,
          "SelectionStatus": "string",
          "Text": "string",
          "TextType": "string"
       }
    ],
    "DetectDocumentTextModelVersion": "string",
    "DocumentMetadata": {
       "Pages": number
    },
    "JobStatus": "string",
    "NextToken": "string",
    "StatusMessage": "string",
    "Warnings": [
       {
          "ErrorCode": "string",
          "Pages": [ number ]
       }
    ]
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

Blocks

文本檢測操作的結果。

類型：的陣列Block對象

DetectDocumentTextModelVersion

類型：字串

DocumentMetadata

有關 Amazon Textract 處理的文檔的信息。DocumentMetadata會在 Amazon Textract 視頻操作的分頁響應的每一頁中返回。

類型：DocumentMetadata 物件

JobStatus

文字偵測任務的目前狀態。

類型：字串

有效值： IN_PROGRESS | SUCCEEDED | FAILED | PARTIAL_SUCCESS

NextToken

如果響應被截斷，Amazon Textract 將返回此令牌。您可以在後續請求中使用此符記來檢索下一組文字偵測結果。

類型：字串

長度限制：長度下限為 1。長度上限為 255。

模式：`.*\S.*`

StatusMessage

如果無法完成檢測任務，則返回。包含有關發生什麼錯誤的說明。

類型：字串

Warnings

文檔的文本檢測操作過程中發生的警告列表。

類型：的陣列Warning對象

錯誤

AccessDeniedException

您未獲授權執行動作。使用已獲授權之使用者或 IAM 角色的 Amazon Resource Name (ARN) 來執行操作。

HTTP 狀態碼：400

InternalServerError

Amazon Textract 發生服務問題。請再次嘗試呼叫。

HTTP 狀態碼：500

InvalidJobIdException

將無效的作業標識符傳遞給GetDocumentAnalysis或GetDocumentAnalysis。

HTTP 狀態碼：400

InvalidKMSKeyException

表示您沒有使用輸入的 KMS 密鑰進行解密權限，或者 KMS 密鑰輸入錯誤。

HTTP 狀態碼：400

InvalidParameterException

輸入參數違反限制。例如，在同步操作中，`InvalidParameterException`異常發生時，`S3Object`或者`Bytes`值提供在`Document`請求參數。請驗證您的參數，然後再次呼叫 API 操作。

HTTP 狀態碼：400

InvalidS3ObjectException

Amazon Textract 無法存取請求中指定的 S3 物件。有關更多信息，請設定 Amazon S3 的存取如需故障診斷資訊，請參閱 。故障診斷 Amazon S3

HTTP 狀態碼：400

ProvisionedThroughputExceededException

請求數超過您的傳輸量限制。如果您希望提高此限制，請聯絡 Amazon Textract。

HTTP 狀態碼：400

ThrottlingException

Amazon Textract 暫時無法處理請求。請再次嘗試呼叫。

HTTP 狀態碼：500

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- AWS Command Line Interface
- 適用於 .NET 的 AWS 開發套件
- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- AWS適用於 Java 的開發套件第 2 版
- 適用於 JavaScript 的 AWS 開發套件
- 適用於 PHP V3 的 AWS 開發套件
- 適用於 Python 的 AWS 開發套件
- 適用於 Ruby V3 的 AWS 開發套件

GetExpenseAnalysis

獲取分析發票和收據的 Amazon Textract 異步操作的結果。Amazon Textract 從輸入的發票和收據中查找聯繫信息、購買的商品和供應商名稱。

啟動異步發票/收據分析，方法是調用StartExpenseAnalysis，它返回一個作業標識符（JobId。在完成發票/收據分析後，Amazon Textract 會將完成狀態發佈到 Amazon Simple Notification Service (Amazon SNS) 主題。此主題必須在初始調用StartExpenseAnalysis。若要取得發票/收據分析操作結果，首先請確認 Amazon SNS 主題的狀態值為SUCCEEDED。如果是這樣，請調用GetExpenseAnalysis，並傳遞作業標識符（JobId）從初始調用到StartExpenseAnalysis。

使用 MaxResults 參數來限制傳回的塊數量。如果結果多於MaxResults，值NextToken在操作響應中包含用於取得下一組結果的字符。若要取得下一頁的結果，請調用GetExpenseAnalysis，然後填入NextToken請求參數與上一次調用返回的令牌值GetExpenseAnalysis。

如需詳細資訊，請參閱「」分析發票和收款。

請求語法

```
{
   "JobId": "string",
   "MaxResults": number,
   "NextToken": "string"
}
```

請求參數

請求接受採用 JSON 格式的下列資料。

JobId

文本檢測任務的唯一識別碼。所以此JobId從傳回StartExpenseAnalysis。一個JobId值只能保留 7 天。

類型：字串

限制：長度下限為 1。長度上限為 64。

模式：^[a-zA-Z0-9-_]+$

：必要 是

MaxResults

每次分頁呼叫可回傳結果的數量上限。您可以指定的最大值為 20。如果您指定的值大於 20，最多只能傳回 20 個結果。預設值為 20。

類型：整數

有效範圍：最小值為 1。

：必要 否

NextToken

如果之前的響應不完整（因為有更多要檢索的塊），Amazon Textract 會在響應中返回一個分頁令牌。您可以使用此分頁符記來檢索下一組塊。

類型：字串

限制：長度下限為 1。長度上限為 255。

模式：`.*\S.*`

：必要 否

回應語法

```
{
   "AnalyzeExpenseModelVersion": "string",
   "DocumentMetadata": {
      "Pages": number
   },
   "ExpenseDocuments": [
      {
         "ExpenseIndex": number,
         "LineItemGroups": [
            {
               "LineItemGroupIndex": number,
               "LineItems": [
                  {
                     "LineItemExpenseFields": [
                        {
                           "LabelDetection": {
                              "Confidence": number,
                              "Geometry": {
```

```
                        "BoundingBox": {
                           "Height": number,
                           "Left": number,
                           "Top": number,
                           "Width": number
                        },
                        "Polygon": [
                           {
                              "X": number,
                              "Y": number
                           }
                        ]
                     },
                     "Text": "string"
                  },
                  "PageNumber": number,
                  "Type": {
                     "Confidence": number,
                     "Text": "string"
                  },
                  "ValueDetection": {
                     "Confidence": number,
                     "Geometry": {
                        "BoundingBox": {
                           "Height": number,
                           "Left": number,
                           "Top": number,
                           "Width": number
                        },
                        "Polygon": [
                           {
                              "X": number,
                              "Y": number
                           }
                        ]
                     },
                     "Text": "string"
                  }
               }
            ]
         }
      ]
   }
],
```

```
        "SummaryFields": [
           {
              "LabelDetection": {
                 "Confidence": number,
                 "Geometry": {
                    "BoundingBox": {
                       "Height": number,
                       "Left": number,
                       "Top": number,
                       "Width": number
                    },
                    "Polygon": [
                       {
                          "X": number,
                          "Y": number
                       }
                    ]
                 },
                 "Text": "string"
              },
              "PageNumber": number,
              "Type": {
                 "Confidence": number,
                 "Text": "string"
              },
              "ValueDetection": {
                 "Confidence": number,
                 "Geometry": {
                    "BoundingBox": {
                       "Height": number,
                       "Left": number,
                       "Top": number,
                       "Width": number
                    },
                    "Polygon": [
                       {
                          "X": number,
                          "Y": number
                       }
                    ]
                 },
                 "Text": "string"
              }
           }
```

```
            ]
         }
      ],
      "JobStatus": "string",
      "NextToken": "string",
      "StatusMessage": "string",
      "Warnings": [
         {
            "ErrorCode": "string",
            "Pages": [ number ]
         }
      ]
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

[AnalyzeExpenseModelVersion](#)

分析費用的當前模型版本。

類型：字串

[DocumentMetadata](#)

有關 Amazon Textract 處理的文檔的信息。`DocumentMetadata`會在 Amazon Textract 操作的分頁響應的每一頁中返回。

類型：[DocumentMetadata](#) 物件

[ExpenseDocuments](#)

Amazon Textract 檢測到的費用。

類型：的陣列[ExpenseDocument](#)對象

[JobStatus](#)

文本檢測任務的目前狀態。

類型：字串

有效值： IN_PROGRESS | SUCCEEDED | FAILED | PARTIAL_SUCCESS

NextToken

如果響應被截斷，Amazon Textract 將返回此令牌。您可以在後續請求中使用此符記來檢索下一組文本檢測結果。

類型：字串

限制：長度下限為 1。長度上限為 255。

模式：.*\S.*

StatusMessage

如果無法完成檢測任務，則返回。包含有關發生什麼錯誤的說明。

類型：字串

Warnings

文檔的文本檢測操作過程中發生的警告列表。

類型：的陣列Warning對象

錯誤

AccessDeniedException

您未獲授權執行動作。使用已獲授權之使用者或 IAM 角色的 Amazon Resource Name (ARN) 來執行操作。

HTTP 狀態碼：400

InternalServerError

Amazon Textract 發生服務問題。請再次嘗試呼叫。

HTTP 狀態碼：500

InvalidJobIdException

將無效的作業標識符傳遞給GetDocumentAnalysis或GetDocumentAnalysis。

HTTP 狀態碼：400

InvalidKMSKeyException

表示您沒有使用輸入的 KMS 密鑰進行解密權限，或者 KMS 密鑰輸入錯誤。

HTTP 狀態碼：400

InvalidParameterException

輸入參數違反限制。例如，在同步操作中，`InvalidParameterException`異常發生時，`S3Object`或者`Bytes`值提供在`Document`請求參數。請驗證您的參數，然後再次呼叫 API 操作。

HTTP 狀態碼：400

InvalidS3ObjectException

Amazon Textract 無法存取請求中指定的 S3 物件。有關詳細信息，請設定對 Amazon S3 的存取如需故障診斷資訊，請參閱 。故障診斷 Amazon S3

HTTP 狀態碼：400

ProvisionedThroughputExceededException

請求數超過您的傳輸量限制。如果您想要提高此限制，請聯絡 Amazon Textract。

HTTP 狀態碼：400

ThrottlingException

Amazon Textract 暫時無法處理請求。請再次嘗試呼叫。

HTTP 狀態碼：500

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- AWS Command Line Interface
- 適用於 .NET 的 AWS 開發套件
- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- AWS適用於 Java 的開發套件第 2 版
- 適用於 JavaScript 的 AWS 開發套件

- 適用於 PHP V3 的 AWS 開發套件
- 適用於 Python 的 AWS 開發套件
- 適用於 Ruby V3 的 AWS 開發套件

StartDocumentAnalysis

開始對輸入檔的異步分析，以瞭解檢測到的項目之間的關係，例如鍵值組、表格和選擇元素。

StartDocumentAnalysis可以分析 JPEG、PNG、TIFF 和 PDF 格式的文檔中的文本。這些文件存放在 Amazon S3 儲存貯體中。使用DocumentLocation指定文件的儲存貯體與影片檔名稱。

StartDocumentAnalysis返回一個作業標識符 (JobId），您可以用來獲取操作的結果。文字分析完成後，Amazon Textract 會將完成狀態態碼發佈至您在NotificationChannel。要獲取文本分析操作的結果，請首先檢查發佈到 Amazon SNS 主題的狀態值是否為SUCCEEDED。如果是這樣，請調用GetDocumentAnalysis，並傳遞作業標識符（JobId）從初始調用到StartDocumentAnalysis。

如需詳細資訊，請參閱「」文件文字分析。

請求語法

```
{
   "ClientRequestToken": "string",
   "DocumentLocation": {
      "S3Object": {
         "Bucket": "string",
         "Name": "string",
         "Version": "string"
      }
   },
   "FeatureTypes": [ "string" ],
   "JobTag": "string",
   "KMSKeyId": "string",
   "NotificationChannel": {
      "RoleArn": "string",
      "SNSTopicArn": "string"
   },
   "OutputConfig": {
      "S3Bucket": "string",
      "S3Prefix": "string"
   }
}
```

請求參數

請求接受採用 JSON 格式的下列資料。

ClientRequestToken

用於標識啟動請求的冪等令牌。如果您將相同的令牌與多個StartDocumentAnalysis請求，相同JobId將傳回。使用ClientRequestToken以防止同一作業意外多次啟動。如需詳細資訊，請參閱「」調用 Amazon Textract 異步操作。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：^[a-zA-Z0-9-_]+$

：必要 否

DocumentLocation

要處理的文檔的位置。

類型：DocumentLocation 物件

：必要 是

FeatureTypes

要執行的分析類型的列表。將 TABLE 添加到列表以返回有關輸入文檔中檢測到的表的信息。添加表單以返回檢測到的表單數據。要執行這兩種類型的分析，請將表和表格添加到FeatureTypes。文檔中檢測到的所有行和單詞都包含在響應中（包括與FeatureTypes。

類型：字串陣列

有效值： TABLES | FORMS

：必要 是

JobTag

您指定的標識符，該編碼包含在發佈到 Amazon SNS 主題的完成通知中。例如，您可以使用JobTag標識完成通知對應的單據類型（例如納稅表或收據）。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：[a-zA-Z0-9_.\-:]+

：必要 否

KMSKeyId

用來加密推斷結果的 KMS 金鑰。這可以是密鑰 ID 或密鑰別名格式。提供 KMS 密鑰後，KMS 密鑰將用於客戶存儲桶中的對象的服務器端加密。如果未啟用此參數，則結果將使用 SSE-S3 加密服務器端。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`^[A-Za-z0-9][A-Za-z0-9:_/+=,@.-]{0,2048}$`

：必要 否

NotificationChannel

您希望亞馬 Amazon Textract 將操作的完成狀態發佈到的 Amazon SNS 主題 ARN。

類型：NotificationChannel 物件

：必要 否

OutputConfig

設置輸出是否轉到客戶定義的時段。默認情況下，Amazon Textract 系統會在內部保存結果，以供 GetDocumentAnalysis 操作訪問。

類型：OutputConfig 物件

：必要 否

回應語法

```
{
   "JobId": "string"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

JobId

文件文字偵測任務的識別碼。使用JobId在後續調用中標識作業GetDocumentAnalysis。一個JobId值僅在 7 天內有效。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：^[a-zA-Z0-9-_]+$

錯誤

AccessDeniedException

您未獲授權執行動作。使用已獲授權之使用者或 IAM 角色的 Amazon Resource Name (ARN) 來執行操作。

HTTP 狀態碼：400

BadDocumentException

Amazon Textract 無法閱讀該文檔。有關 Amazon Textract 中文檔限制的更多信息，請參閱亞馬遜文字中的硬性限制。

HTTP 狀態碼：400

DocumentTooLargeException

無法處理該文檔，因為它太大。同步操作的最大文件大小為 10 MB。對於 PDF 文件，異步操作的最大文檔大小為 500 MB。

HTTP 狀態碼：400

IdempotentParameterMismatchException

一個ClientRequestToken輸入參數重複用於一個操作，但至少有一個其他輸入參數不同於先前對操作的呼叫。

HTTP 狀態碼：400

InternalServerError

Amazon Textract 發生服務問題。請再次嘗試呼叫。

HTTP 狀態碼：500

InvalidKMSKeyException

表示您沒有使用輸入的 KMS 密鑰進行解密權限，或者 KMS 密鑰輸入錯誤。

HTTP 狀態碼：400

InvalidParameterException

輸入參數違反限制。例如，在同步操作中，InvalidParameterException異常發生時，S3Object或者Bytes值提供在Document請求參數。請驗證您的參數，然後再次呼叫 API 操作。

HTTP 狀態碼：400

InvalidS3ObjectException

Amazon Textract 無法存取請求中指定的 S3 物件。有關詳細信息，請設定對 Amazon S3 的存取如需故障診斷資訊，請參閱 。故障診斷 Amazon S3

HTTP 狀態碼：400

LimitExceededException

超出 Amazon Textract 服務限制。例如，如果您同時啟動太多的異步作業，則調用以啟動操作 (StartDocumentTextDetection) 將引發 LimitExceededException 制 (HTTP 狀態碼：400)，直到數量同時執行任務的數量低於 Amazon Textract 服務限制。

HTTP 狀態碼：400

ProvisionedThroughputExceededException

請求數超過您的傳輸量限制。如果您希望提高此限制，請聯絡 Amazon Textract。

HTTP 狀態碼：400

ThrottlingException

Amazon Textract 暫時無法處理請求。請再次嘗試呼叫。

HTTP 狀態碼：500

UnsupportedDocumentException

不支持輸入檔的格式。操作文檔可以採用 PNG、JPEG、PDF 或 TIFF 格式。

HTTP 狀態碼：400

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- AWS Command Line Interface
- 適用於 .NET 的 AWS 開發套件
- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- AWS適用於 Java 的開發套件第 2 版
- 適用於 JavaScript 的 AWS 開發套件
- 適用於 PHP V3 的 AWS 開發套件
- 適用於 Python 的 AWS 開發套件
- 適用於 Ruby V3 的 AWS 開發套件

StartDocumentTextDetection

開始非同步偵測文件中的文字。Amazon Textract 可以檢測文本行和構成一行文本的單詞。

StartDocumentTextDetection可以分析 JPEG、PNG、TIFF 和 PDF 格式的文檔中的文本。這些文件存放在 Amazon S3 儲存貯體中。使用DocumentLocation指定文件的儲存貯體與影片檔名稱。

StartTextDetection返回一個作業標識符 (JobId），用於獲取操作結果。文字偵測完成後，Amazon Textract 會將完成狀態發佈至 Amazon Simple Notification Service (Amazon SNS) 主題的完成狀態，而此主題會傳送至您在NotificationChannel。要獲取文本檢測操作的結果，請首先檢查發佈到 Amazon SNS 主題的狀態值是SUCCEEDED。如果是這樣，請調用GetDocumentTextDetection，並傳遞作業標識符（JobId）從初始調用到StartDocumentTextDetection。

如需詳細資訊，請參閱「」文件文字偵測。

請求語法

```
{
   "ClientRequestToken": "string",
   "DocumentLocation": {
      "S3Object": {
         "Bucket": "string",
         "Name": "string",
         "Version": "string"
      }
   },
   "JobTag": "string",
   "KMSKeyId": "string",
   "NotificationChannel": {
      "RoleArn": "string",
      "SNSTopicArn": "string"
   },
   "OutputConfig": {
      "S3Bucket": "string",
      "S3Prefix": "string"
   }
}
```

請求參數

請求接受採用 JSON 格式的下列資料。

ClientRequestToken

用於標識啟動請求的冪等令牌。如果您將相同的令牌與多個StartDocumentTextDetection請求，相同JobId傳回。使用ClientRequestToken以防止同一作業意外多次啟動。如需詳細資訊，請參閱「」調用 Amazon Textract 異步操作。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：^[a-zA-Z0-9-_]+$

：必要 否

DocumentLocation

要處理的文檔的位置。

類型：DocumentLocation 物件

：必要 是

JobTag

您指定的標識符，該編碼包含在發佈到 Amazon SNS 主題的完成通知中。例如，您可以使用JobTag標識完成通知對應的單據類型（例如納稅表或收據）。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：[a-zA-Z0-9_.\-:]+

：必要 否

KMSKeyId

用於加密推斷結果的 KMS 金鑰。此格式可以是密鑰 ID 或密鑰別名格式。提供 KMS 密鑰後，KMS 密鑰將用於客戶存儲桶中的對象的服務器端加密。如果未啟用此參數，則結果將使用 SSE-S3 加密服務器端。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：^[A-Za-z0-9][A-Za-z0-9:_/+=,@.-]{0,2048}$

：必要 否

NotificationChannel

您希望亞馬 Amazon Textract 將操作的完成狀態發佈到的 Amazon SNS 主題 ARN。

類型：NotificationChannel 物件

：必要 否

OutputConfig

設置輸出是否轉到客户定義的存儲段。默認情況下，Amazon Textract 將在內部保存結果，以便通過獲取 GetDocumentTextDetection 操作進行訪問。

類型：OutputConfig 物件

：必要 否

回應語法

```
{
   "JobId": "string"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

JobId

文檔的文本檢測作業的標識符。使用JobId在後續調用中標識作業GetDocumentTextDetection。一個JobId值僅在 7 天內有效。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：^[a-zA-Z0-9-_]+$

錯誤

AccessDeniedException

您未獲授權執行動作。使用已獲授權之使用者或 IAM 角色的 Amazon Resource Name (ARN) 來執行操作。

HTTP 狀態碼：400

BadDocumentException

Amazon Textract 無法閱讀該文檔。有關 Amazon Textract 中文檔限制的更多信息，請參閱亞馬遜文字中的硬性限制。

HTTP 狀態碼：400

DocumentTooLargeException

無法處理該文檔，因為它太大。同步操作的最大文件大小為 10 MB。對於 PDF 文件，異步操作的最大文檔大小為 500 MB。

HTTP 狀態碼：400

IdempotentParameterMismatchException

一個`ClientRequestToken`輸入參數與操作一起重複用，但至少有一個其他輸入參數不同於先前對操作的呼叫。

HTTP 狀態碼：400

InternalServerError

Amazon Textract 發生服務問題。請再次嘗試呼叫。

HTTP 狀態碼：500

InvalidKMSKeyException

表示您沒有使用輸入的 KMS 密鑰進行解密權限，或者 KMS 密鑰輸入錯誤。

HTTP 狀態碼：400

InvalidParameterException

輸入參數違反限制。例如，在同步操作中，`InvalidParameterException`異常發生時，`S3Object`或者`Bytes`值提供在`Document`請求參數。請驗證您的參數，然後再次呼叫 API 操作。

HTTP 狀態碼：400

InvalidS3ObjectException

Amazon Textract 無法存取請求中指定的 S3 物件。有關詳細信息，請設定對 Amazon S3 的存取如需故障診斷資訊，請參閱 。故障診斷 Amazon S3

HTTP 狀態碼：400

LimitExceededException

超出 Amazon Textract 服務限制。例如，如果您同時啟動太多的異步作業，則調用以啟動操作 (`StartDocumentTextDetection`) 引發限制拒絕例外狀況 (HTTP 狀態碼：400)，直到數量同時執行任務的數量低於 Amazon Textract 服務限制。

HTTP 狀態碼：400

ProvisionedThroughputExceededException

請求數超過您的傳輸量限制。如果您希望提高此限制，請聯絡 Amazon Textract。

HTTP 狀態碼：400

ThrottlingException

Amazon Textract 暫時無法處理請求。請再次嘗試呼叫。

HTTP 狀態碼：500

UnsupportedDocumentException

不支持輸入檔案的格式。操作文檔可以採用 PNG、JPEG、PDF 或 TIFF 格式。

HTTP 狀態碼：400

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- AWS Command Line Interface
- 適用於 .NET 的 AWS 開發套件
- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- AWS適用於 Java 的開發套件第 2 版

- 適用於 JavaScript 的 AWS 開發套件
- 適用於 PHP V3 的 AWS 開發套件
- 適用於 Python 的 AWS 開發套件
- 適用於 Ruby V3 的 AWS 開發套件

StartExpenseAnalysis

開始對聯繫信息、購買物料和供應商名稱等數據的發票或收據進行異步分析。

StartExpenseAnalysis可以分析 JPEG、PNG 和 PDF 格式的文檔中的文本。文件必要儲存在 Amazon S3 儲存貯體中。使用DocumentLocation參數指定您的 S3 儲存貯體名稱以及該儲存貯體中檔案的名稱。

StartExpenseAnalysis返回一個作業標識符 (JobId），您將提供給GetExpenseAnalysis檢索操作的結果。完成對輸入發票/收據的分析後，Amazon Textract 會向您提供的亞馬遜 Simple Notification Service (Amazon SNS) 主題發佈完成狀態。NotificationChannel。要獲得發票和收據分析操作的結果，請確保發佈到 Amazon SNS 主題的狀態值為SUCCEEDED。如果是這樣，請調用GetExpenseAnalysis，並傳遞作業標識符（JobId），您通過調用StartExpenseAnalysis。

如需詳細資訊，請參閱「」分析發票和收款。

請求語法

```
{
   "ClientRequestToken": "string",
   "DocumentLocation": {
      "S3Object": {
         "Bucket": "string",
         "Name": "string",
         "Version": "string"
      }
   },
   "JobTag": "string",
   "KMSKeyId": "string",
   "NotificationChannel": {
      "RoleArn": "string",
      "SNSTopicArn": "string"
   },
   "OutputConfig": {
      "S3Bucket": "string",
      "S3Prefix": "string"
   }
}
```

請求參數

請求接受採用 JSON 格式的下列資料。

ClientRequestToken

用於標識啟動請求的冪等令牌。如果您將相同的令牌與多個StartDocumentTextDetection請求，相同JobId傳回。使用ClientRequestToken以防止同一作業意外多次啟動。如需詳細資訊，請參閱「」調用 Amazon Textract 異步操作

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：^[a-zA-Z0-9-_]+$

：必要 否

DocumentLocation

要處理的文檔的位置。

類型：DocumentLocation 物件

：必要 是

JobTag

您指定的標識符，該編碼包含在發佈到 Amazon SNS 主題的完成通知中。例如，您可以使用JobTag標識完成通知對應的單據類型（例如納稅表或收據）。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：[a-zA-Z0-9_.\-:]+

：必要 否

KMSKeyId

用來加密推理結果的 KMS 金鑰。此格式可以是密鑰 ID 或密鑰別名格式。提供 KMS 密鑰後，KMS 密鑰將用於客户存儲桶中的對象的服務器端加密。如果未啟用此參數，則結果將使用 SSE-S3 加密服務器端。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：^[A-Za-z0-9][A-Za-z0-9:_/+=,@.-]{0,2048}$

：必要 否

NotificationChannel

您希望亞馬 Amazon Textract 將操作的完成狀態發佈到的 Amazon SNS 主題 ARN。

類型：NotificationChannel 物件

：必要 否

OutputConfig

設置輸出是否轉到客户定義的存儲段。默認情況下，Amazon Textract 會在內部保存結果，以便GetExpenseAnalysisoperation.

類型：OutputConfig 物件

：必要 否

回應語法

```
{
   "JobId": "string"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

JobId

文本檢測任務的唯一識別符。所以此JobId從傳回StartExpenseAnalysis。一個JobId值僅在 7 天內有效。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：^[a-zA-Z0-9-_]+$

錯誤

AccessDeniedException

您未獲授權執行動作。使用已獲授權之使用者或 IAM 角色的 Amazon Resource Name (ARN) 來執行操作。

HTTP 狀態碼：400

BadDocumentException

Amazon Textract 無法閱讀該文檔。有關 Amazon Textract 中文檔限制的更多信息，請參閱亞馬遜文字中的硬性限制。

HTTP 狀態碼：400

DocumentTooLargeException

無法處理該文檔，因為它太大。同步操作的最大檔案大小為 10 MB。對於 PDF 文件，異步操作的最大文檔大小為 500 MB。

HTTP 狀態碼：400

IdempotentParameterMismatchException

一個`ClientRequestToken`輸入參數重複用於一個操作，但至少有一個其他輸入參數不同於先前對操作的呼叫。

HTTP 狀態碼：400

InternalServerError

Amazon Textract 發生服務問題。請再次嘗試呼叫。

HTTP 狀態碼：500

InvalidKMSKeyException

表示您沒有使用輸入的 KMS 密鑰進行解密權限，或者 KMS 密鑰輸入錯誤。

HTTP 狀態碼：400

InvalidParameterException

輸入參數違反限制。例如，在同步操作中，`InvalidParameterException`異常發生時，`S3Object`或者`Bytes`值提供在`Document`請求參數。請驗證您的參數，然後再次呼叫 API 操作。

HTTP 狀態碼：400

InvalidS3ObjectException

Amazon Textract 無法存取請求中指定的 S3 物件。有關詳細信息，請設定對 Amazon S3 的存取如需故障診斷資訊，請參閱 。故障診斷 Amazon S3

HTTP 狀態碼：400

LimitExceededException

超出 Amazon Textract 服務限制。例如，如果您同時啟動太多的異步作業，則調用以啟動操作 (`StartDocumentTextDetection`) 引發限制拒絕例外狀況 (HTTP 狀態碼：400)，直到數量同時執行任務的數量低於 Amazon Textract 服務限制。

HTTP 狀態碼：400

ProvisionedThroughputExceededException

請求數超過您的傳輸量限制。如果您希望提高此限制，請聯絡 Amazon Textract。

HTTP 狀態碼：400

ThrottlingException

Amazon Textract 暫時無法處理請求。請再次嘗試呼叫。

HTTP 狀態碼：500

UnsupportedDocumentException

輸入檔案的格式不受支持。操作文檔可以採用 PNG、JPEG、PDF 或 TIFF 格式。

HTTP 狀態碼：400

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- AWS Command Line Interface
- 適用於 .NET 的 AWS 開發套件
- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- AWS適用於 Java 的開發套件第 2 版

- 適用於 JavaScript 的 AWS 開發套件
- 適用於 PHP V3 的 AWS 開發套件
- 適用於 Python 的 AWS 開發套件
- 適用於 Ruby V3 的 AWS 開發套件

資料類型

目前支援下列資料類型：

- AnalyzeIDDetections
- Block
- BoundingBox
- Document
- DocumentLocation
- DocumentMetadata
- ExpenseDetection
- ExpenseDocument
- ExpenseField
- ExpenseType
- Geometry
- HumanLoopActivationOutput
- HumanLoopConfig
- HumanLoopDataAttributes
- IdentityDocument
- IdentityDocumentField
- LineItemFields
- LineItemGroup
- NormalizedValue
- NotificationChannel
- OutputConfig
- Point

- Relationship
- S3Object
- Warning

AnalyzeIDDetections

用於包含分析 Id 操作檢測到的信息。

內容

Confidence

偵測到的文字置信度。

類型：Float

有效範圍：最小值為 0。最大值為 100。

：必要 否

NormalizedValue

僅返回日期，返回檢測到的值的類型和以更具機器可讀性的方式寫入的日期。

類型：NormalizedValue 物件

：必要 否

Text

標準化字段或與其關聯的值的文本。

類型：字串

：必要 是

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- AWS適用於 Java 的開發套件第 2 版
- 適用於 Ruby V3 的 AWS 開發套件

Block

一個Block表示在文檔中識別的一組相互接近的像素內的項目。返回的信息在Block物件取決於操作類型。在文檔的文本檢測中（例如DetectDocumentText），您可以獲得有關檢測到的單詞和文本行的信息。在文本分析中（例如AnalyzeDocument），您還可以獲取有關文檔中檢測到的字段、表格和選擇元素的信息。

陣列Block對象由同步操作和異步操作返回。在同步操作中，例如DetectDocumentText，陣列Block對象是整個結果集。在異步操作中，例如GetDocumentAnalysis時，數組將通過一個或多個響應返回。

如需詳細資訊，請參閱「」Amazon Textract 運作方式。

內容

BlockType

已識別的文本項的類型。在文本檢測操作中，返回以下類型：

- 頁面-包含線的清單Block在文檔頁面上檢測到的對象。
- 字-在文檔頁面上檢測到的單詞。單字是一或多個 ISO 基本拉丁指令碼字元，不以空格分隔。
- 線-在文檔頁面上檢測到的製表符分隔的連續單詞的字符串。

在文本分析操作中，返回以下類型：

- 頁面-包含子項目Block在文檔頁面上檢測到的對象。
- 鍵值集-存儲密鑰和值Block對象，用於在文檔頁面上檢測到的鏈接文本。使用EntityType字段來確定鍵值集對象是否為密鑰Block對象或值Block物件。
- 字-在文檔頁面上檢測到的單詞。單字是一或多個 ISO 基本拉丁指令碼字元，不以空格分隔。
- 線-在文檔頁面上檢測到的製表符分隔的連續單詞的字符串。
- 表-在文檔頁面上檢測到的表。表是具有兩個或多個行或列的基於網格的信息，單元格跨度包含一行和一列。
- 細胞-檢測到的表中的單元格。單元格是包含單元格中文本的塊的父項。
- 選擇元素-選擇元素，例如選項按鈕（單選按鈕）或在文檔頁面上檢測到的複選框。使用SelectionStatus，以判斷選取元素的狀態。

類型：字串

有效值： KEY_VALUE_SET | PAGE | LINE | WORD | TABLE | CELL | SELECTION_ELEMENT

：必要 否

ColumnIndex

在其中顯示表單元格的列。第一列位置為 1。ColumnIndex不返回DetectDocumentText和GetDocumentTextDetection。

類型：整數

有效範圍：最小值為 0。

：必要 否

ColumnSpan

表單元格跨越的列數。目前，此值始終為 1，即使跨越的列數大於 1。ColumnSpan不返回DetectDocumentText和GetDocumentTextDetection。

類型：整數

有效範圍：最小值為 0。

：必要 否

Confidence

Amazon Textract 在識別文本的準確性以及識別文本周圍幾何點的準確性方面的置信度得分。

類型：Float

有效範圍：最小值為 0。最大值為 100。

：必要 否

EntityTypes

實體類型。可能返回以下內容：

- 鍵-文檔上某個字段的標識符。
- 值-字段文本。

EntityTypes不返回DetectDocumentText和GetDocumentTextDetection。

類型：字串陣列

有效值： KEY | VALUE

：必要 否

Geometry

已識別文字在圖像上的位置。它包括一個圍繞文本的軸對齊粗邊界框，以及一個細粒度的多邊形，以獲得更精確的空間信息。

類型：Geometry 物件

：必要 否

Id

識別文本的標識符。標識符僅對於單個操作是唯一的。

類型：字串

模式：`.*\S.*`

：必要 否

Page

檢測到塊的頁面。Page由異步操作返回。僅為 PDF 或 TIFF 格式的多頁文檔返回大於 1 的頁面值。掃描的圖像 (JPEG/PNG)，即使它包含多個文檔頁面，也被視為單頁文檔。的值為Page一律為 1。同步操作不返回Page因為每個輸入文檔都被視為單頁文檔。

類型：整數

有效範圍：最小值為 0。

：必要 否

Relationships

當前塊子塊的列表。例如，LINE 對象具有作為文本行一部分的每個 WORD 塊的子塊。列表中沒有不存在的關係對象，例如當前塊沒有子塊時。列表大小可以如下：

- 0-塊沒有子塊。
- 1-該塊具有子塊。

類型：陣列Relationship對象

：必要 否

RowIndex

表單元格所在的行。第一行位置為 1。RowIndex不返回DetectDocumentText和GetDocumentTextDetection。

類型：整數

有效範圍：最小值為 0。

：必要 否

RowSpan

表單元格跨越的行數。目前，此值始終為 1，即使跨越的行數大於 1。RowSpan不返回DetectDocumentText和GetDocumentTextDetection。

類型：整數

有效範圍：最小值為 0。

：必要 否

SelectionStatus

選擇元素的選擇狀態，例如選項按鈕或複選框。

類型：字串

有效值： SELECTED | NOT_SELECTED

：必要 否

Text

Amazon Textract 識別的單詞或文本行。

類型：字串

：必要 否

TextType

Amazon Textract 檢測到的文本類型。可以檢查是否有手寫文字和打印文本。

類型：字串

有效值： `HANDWRITING | PRINTED`

：必要 否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- AWS適用於 Java 的開發套件第 2 版
- 適用於 Ruby V3 的 AWS 開發套件

BoundingBox

文檔頁面上檢測到的頁面、文本、鍵值對、表格、表格單元格或選擇元素周圍的邊界框。所以此`left`（x 座標）和`top`（y 軸）是代表上方和左邊方的座標。請注意，影像的左上角是原點 (0,0)。

所以此`top`和`left`返回的值是整個文檔頁面大小的比率。例如，如果輸入影像為 700 x 200 像素，而週框方塊的左上方座標為 350 x 50 像素，則 API 會傳回 `left` 值 0.5 (350/700) 和 `top` 值 0.25 (50/200)。

所以此`width`和`height`值以整體文檔頁面維度的比例表示，以整體文檔頁面維度的比例表示。例如，如果文檔頁面大小為 700 x 200 像素，邊界框寬度為 70 像素，則返回的寬度為 0.1。

內容

Height

高度方塊的高度，以整體文檔頁面高度的比例表示。

類型：Float

：必要 否

Left

左邊方塊的左邊座標，以整體文檔頁面寬度的比例表示。

類型：Float

：必要 否

Top

上方方座標，以整體文檔頁面高度的比例表示。

類型：Float

：必要 否

Width

寬度方塊的寬度，以整體文檔頁面寬度的比例表示。

類型：Float

：必要 否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- AWS適用於 Java 的開發套件第 2 版
- 適用於 Ruby V3 的 AWS 開發套件

Document

輸入文檔，以字節形式或 S3 對象形式。

您可以將影像位元組傳遞至 Amazon Textract 章 API 操作，使用Bytes屬性。例如，您需要使用Bytes屬性傳遞從本地文件系統加載的文檔。圖像字節通過使用Bytes屬性必須經過 base64 編碼。如果您使用 AWS 開發工具包調用 Amazon Textract API 操作，則您的代碼可能無需對文檔文件字節進行編碼。

您可以將 S3 儲存貯體中的影像傳遞至 Amazon Textract API 操作，使用S3Object屬性。存放在 S3 儲存貯體中的文檔，不需要 base64 編碼。

含有 S3 物件的 S3 儲存貯體區域必須符合您用於 Amazon Textract 操作的 AWS 區域。

如果您使用 AWS CLI 呼叫 Amazon Textract 操作，則不支援使用位元組來傳遞。您必須先將文檔上傳至 Amazon S3 儲存貯體，再使用 S3Object 屬性呼叫操作。

若要使用 Amazon Textract 處理 S3 物件，用户必須具有 S3 物件的存取許可。

內容

Bytes

一個由 base64 編碼的文檔字節組成的 Blob。字節 Blob 中提供的文檔的最大大小為 5 MB。文檔字節必須採用 PNG 或 JPEG 格式。

如果您使用 AWS 開發工具包調用 Amazon Textract，則可能無需使用Bytes欄位。

類型：Base64 編碼二進位數據物件

長度限制：長度下限為 1。長度上限為 104857。

必要 否

S3Object

將 S3 對象標識為文檔源。存儲在 S3 存儲桶中的文檔的最大大小為 5 MB。

類型：S3Object 物件

必要 否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- AWS適用於 Java 的開發套件第 2 版
- 適用於 Ruby V3 的 AWS 開發套件

DocumentLocation

內含待處理檔案的 Amazon S3 儲存儲體。它被異步操作（如StartDocumentTextDetection。

輸入文檔可以是 JPEG 或 PNG 格式的圖像文件。它也可以是 PDF 格式的文件。

內容

S3Object

內含輸入檔案的 Amazon S3 儲存儲體。

類型：S3Object 物件

：必要 否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- AWS適用於 Java 的開發套件第 2 版
- 適用於 Ruby V3 的 AWS 開發套件

DocumentMetadata

有關輸入文檔的信息。

內容

Pages

在文檔中檢測到的頁數。

類型：整數

有效範圍：最小值為 0。

必要：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- AWS適用於 Java 的開發套件第 2 版
- 適用於 Ruby V3 的 AWS 開發套件

ExpenseDetection

用於存儲 Amazon Textract 檢測到的值或標籤相關信息的對象。

內容

Confidence

檢測的可信度，以百分比表示

類型：Float

有效範圍：最小值為 0。最大值為 100。

：必要 否

Geometry

有關以下項目在文件頁面上的位置的信息：檢測到的頁面、文字、鍵值組、表格單元格和選擇元素。

類型：Geometry 物件

：必要 否

Text

Amazon Textract 識別的單詞或文本行

類型：字串

：必要 否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- AWS適用於 Java 的開發套件第 2 版
- 適用於 Ruby V3 的 AWS 開發套件

ExpenseDocument

保存分析費用返回的所有信息的結構

內容

ExpenseIndex

表示信息來自單據中的發票或收據。第一個文檔為 1，第二個文檔為 2，以此類推。

類型：整數

有效範圍：最小值為 0。

：必要 否

LineItemGroups

在文檔的每個表格上檢測到的信息，分別為`LineItems`。

類型：的陣列LineItemGroup對象

：必要 否

SummaryFields

Amazon Textract 在表格之外找到的任何信息。

類型：的陣列ExpenseField對象

：必要 否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- AWS適用於 Java 的開發套件第 2 版
- 適用於 Ruby V3 的 AWS 開發套件

ExpenseField

對檢測到的信息進行細分，分別為災難類型、標籤檢測和價值檢測

內容

LabelDetection

檢測到的元素的明確聲明的標籤。

類型：ExpenseDetection 物件

：必要 否

PageNumber

檢測到值的頁碼。

類型：整數

有效範圍：最小值為 0。

：必要 否

Type

檢測到的元素的隱含標籤。與顯式元素的標籤檢測一起顯示。

類型：ExpenseType 物件

：必要 否

ValueDetection

檢測到的元素的值。存在於顯式和隱式元素中。

類型：ExpenseDetection 物件

：必要 否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件

- 適用於 Go 的 AWS 開發套件
- AWS適用於 Java 的開發套件第 2 版
- 適用於 Ruby V3 的 AWS 開發套件

ExpenseType

用於存儲 Amazon Textract 檢測到的類型相關信息的對象。

內容

Confidence

準確度的置信度，以百分比表示。

類型：Float

有效範圍：最小值為 0。最大值為 100。

使用必要項目 否

Text

Amazon Textract 檢測到的單詞或文本行。

類型：字串

使用必要項目 否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- AWS適用於 Java 的開發套件第 2 版
- 適用於 Ruby V3 的 AWS 開發套件

Geometry

以下項目在文件頁中的位置的信息：檢測到的頁面、文字、鍵值組、表格單元格和選擇元素。

內容

BoundingBox

在文檔頁面上已識別項目位置的軸對齊粗糙表示。

類型：BoundingBox 物件

的必要項目 否

Polygon

在邊界框內，已識別項目周圍的細粒度多邊形。

類型：的陣列Point對象

的必要項目 否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- AWS適用於 Java 的開發套件第 2 版
- 適用於 Ruby V3 的 AWS 開發套件

HumanLoopActivationOutput

顯示循環評估中人類的結果。如果沒有人工解析，則輸入不會觸發人工審查。

內容

HumanLoopActivationConditionsEvaluationResults

顯示狀態評估的結果，包括激活人工審查的條件。

類型：字串

長度限制：長度上限為 10240。

必要項目：否

HumanLoopActivationReasons

顯示是否需要以及為什麼需要人工審查。

類型：字串陣列

陣列成員：項目數下限為 1。

必要項目：否

HumanLoopArn

創建的 HumanLoop Source Name (ARN)。

類型：字串

長度限制：長度上限為 256。

必要項目：否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- AWS適用於 Java 的開發套件第 2 版

- 適用於 Ruby V3 的 AWS 開發套件

HumanLoopConfig

設置文檔在滿足其中一個條件時將發送到的人工審閱工作流程。您還可以在查看之前設置圖像的某些屬性。

內容

DataAttributes

設置輸入資料屬性。

類型：HumanLoopDataAttributes 物件

：必要 否

FlowDefinitionArn

流程定義的 Amazon Resource Name (ARN)。

類型：字串

長度限制：長度上限為 256。

：必要 是

HumanLoopName

用於此圖像的人工作流程名稱。區域中的這一點應保持獨特。

類型：字串

長度限制：長度下限為 1。長度上限為 63。

模式：`^[a-z0-9](-*[a-z0-9])*`

：必要 是

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件

- AWS適用於 Java 的開發套件第 2 版
- 適用於 Ruby V3 的 AWS 開發套件

HumanLoopDataAttributes

允許您設置圖像的屬性。目前，您可以聲明圖片不含個人身份信息和成人內容。

內容

ContentClassifiers

設置輸入圖像是不含個人識別資訊或成人內容。

類型：字串陣列

陣列成員：項目數上限為 256。

有效值： `FreeOfPersonallyIdentifiableInformation | FreeOfAdultContent`

：必要 否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- AWS適用於 Java 的開發套件第 2 版
- 適用於 Ruby V3 的 AWS 開發套件

IdentityDocument

列出在 AnalyzeID 操作中處理的每個文檔的結構。

內容

DocumentIndex

表示文檔在標識文檔列表中的位置。第一個文檔標記為 1，第二個 2，依此類推。

類型：整數

有效範圍：最小值為 0。

必要 否

IdentityDocumentFields

用於記錄從身份證件中提取的信息的結構。包含已提取文本的標準化字段和值。

類型：陣列IdentityDocumentField對象

必要 否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- AWS適用於 Java 的開發套件第 2 版
- 適用於 Ruby V3 的 AWS 開發套件

IdentityDocumentField

包含已提取信息的規範化類型和與其關聯的文本的結構。它們分別提取為類型和值。

內容

Type

用於包含分析 Id 操作檢測到的信息。

類型：AnalyzeIDDetections 物件

：必要 否

ValueDetection

用於包含分析 Id 操作檢測到的信息。

類型：AnalyzeIDDetections 物件

：必要 否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- AWS適用於 Java 的開發套件第 2 版
- 適用於 Ruby V3 的 AWS 開發套件

LineItemFields

保存有關文檔表中不同行的信息的結構。

內容

LineItemExpenseFields

用於顯示表中檢測到的行信息的字段。

類型：陣列ExpenseField對象

：必要 否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- AWS適用於 Java 的開發套件第 2 版
- 適用於 Ruby V3 的 AWS 開發套件

LineItemGroup

包含 LineItems 的表的分組，每個表都由表的`LineItemGroupIndex`。

內容

LineItemGroupIndex

用於標識文檔中特定表的編號。遇到的第一個表格將有一個行項目組索引為 1，第二個 2，依此類推。

類型：整數

有效範圍：最小值為 0。

：必要 否

LineItems

表格的特定行上的信息細分。

類型：陣列為LineItemFields對象

：必要 否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- AWS適用於 Java 的開發套件第 2 版
- 適用於 Ruby V3 的 AWS 開發套件

NormalizedValue

包含與文檔中日期相關的信息，包括值的類型和值。

內容

Value

日期的值，寫為年月-日：分鐘：秒。

類型：字串

必要 否

ValueType

檢測到的值的標準化類型。在這種情況下，日期。

類型：字串

有效值： DATE

必要 否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- AWS適用於 Java 的開發套件第 2 版
- 適用於 Ruby V3 的 AWS 開發套件

NotificationChannel

Amazon Simple Notification Service (Amazon SNS) 主題，Amazon Textract 將異步文檔操作的完成狀態發佈至此主題，例如StartDocumentTextDetection。

內容

RoleArn

IAM 角色的 Amazon Resource Name (ARN)，該角色授予 Amazon Textract 發佈 Amazon SNS 主題的許可。

類型：字串

長度限制：長度下限為 20。長度上限為 2048。

模式：`arn:([a-z\d-]+):iam::\d{12}:role/?[a-zA-Z_0-9+=,.@\-_/]+`

：必要 是

SNSTopicArn

亞馬遜 Textract 將完成狀態發佈到的 Amazon SNS 主題。

類型：字串

長度限制：長度下限為 20。長度上限為 1024。

模式：`(^arn:([a-z\d-]+):sns:[a-zA-Z\d-]{1,20}:\w{12}:.+$)`

：必要 是

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- AWS適用於 Java 的開發套件第 2 版
- 適用於 Ruby V3 的 AWS 開發套件

OutputConfig

設置輸出是否轉到用户創建的存儲桶。用於設置存儲桶的名稱和輸出文件上的前綴。

OutputConfig是一個可選參數，可讓您調整輸出的放置位置。默認情況下，Amazon Textract 將在內部存儲結果，並且只能通過獲取 API 操作訪問。啟用 OutputConfig 後，您可以設置將輸出發送到的存儲桶的名稱以及您可以下載結果的結果的文件前綴。此外，您也可以設置KMSKeyID參數添加到客户主金鑰 (CMK)，以加密您的輸出。如果沒有此參數設置，Amazon Textract 將使用適用於 Amazon S3 的 AWS 託管 CMK 對服務器端進行加密。

Amazon Textract 處理文檔時，必須對買家內容進行解密。如果您的帳户根據 AI 服務退出政策選擇退出，則在服務處理客户內容後，所有未加密的客户內容都會立即永久刪除。Amazon Textract 不會保留輸出的副本。如需如何選擇退出的資訊，請參管理 AI 服務選擇退出政策。

如需資料隱私權的詳細資訊，請參資料私權常見問答。

內容

S3Bucket

輸出要轉到的存儲桶的名稱。

類型：字串

長度限制：長度下限為 3。長度上限為 255。

模式：[0-9A-Za-z\.\-_]*

：必要 是

S3Prefix

物件金鑰的字首，而該物件金鑰是保存輸出的起源。如果未啟用，則前綴將為「文本輸出」。

類型：字串

長度限制：長度下限為 1。長度上限為 1024。

模式：.*\S.*

：必要 否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- AWS適用於 Java 的開發套件第 2 版
- 適用於 Ruby V3 的 AWS 開發套件

Point

文檔頁面上點的 X 和 Y 座標。返回的 X 和 Y 值是整個文檔頁面大小的比率。例如，如果輸入文檔為 700 x 200，且操作返回 X=0.5 且 Y=0.25，則該點位於文檔頁面上 (350,50) 像素座標處。

陣列Point物件,Polygon的一部分，返回為Geometry對象，該對象在Block物件。一個Polygon對象表示檢測到的文本、鍵值對、表格、表格單元格或選擇元素周圍的細粒度多邊形。

內容

X

上某個點的 X 座標值Polygon。

類型：Float

：必要 否

Y

上某個點的 Y 座標值Polygon。

類型：Float

：必要 否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- AWS適用於 Java 的開發套件第 2 版
- 適用於 Ruby V3 的 AWS 開發套件

Relationship

有關塊如何相互關聯的信息。一個Block對象包含 0 或更多Relation列表中的對象，Relationships。如需詳細資訊，請參閱 Block。

所以此Type元素提供了所有塊的關係類型IDs陣列。

內容

Ids

相關塊的 ID 數組。您可以從Type元素。

類型：字串陣列

模式：.*\S.*

：必要 否

Type

ID 數組中的塊與當前塊的關係類型。兩者間的關係可用VALUE或者CHILD。VALUE 類型的關係是一個列表，其中包含與鍵值對的 KEY 相關聯的 VALUE 塊的 ID。類型 CIDD 的關係是一個 ID 的列表，用於標識行單元格塊（表中的單元格塊），在選擇元素的情況下標識 WORD 塊。

類型：字串

有效值： VALUE | CHILD | COMPLEX_FEATURES

：必要 否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- AWS適用於 Java 的開發套件第 2 版
- 適用於 Ruby V3 的 AWS 開發套件

S3Object

標識文檔的 S3 存儲桶名稱和文件名。

包含文檔的 S3 存儲桶的 AWS 區域必須與您用於 Amazon Textract 操作的區域匹配。

為了使 Amazon Textract 能夠處理 S3 存儲桶中的文件，用户必須具有訪問 S3 存儲桶和文件的權限。

內容

Bucket

S3 儲存貯體的名稱。請注意，# 字符在文件名中無效。

類型：字串

長度限制：長度下限為 3。長度上限為 255。

模式：`[0-9A-Za-z\.\-_]*`

必要 否

Name

輸入文檔的文件名。同步操作可以使用 JPEG 或 PNG 格式的圖像文件。異步操作還支持 PDF 和 TIFF 格式文件。

類型：字串

長度限制：長度下限為 1。長度上限為 1024。

模式：`.*\S.*`

必要 否

Version

如果儲存儲體已啟用版本控制，您可以指定物件版本。

類型：字串

長度限制：長度下限為 1。長度上限為 1024。

模式：`.*\S.*`

必要 否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- AWS適用於 Java 的開發套件第 2 版
- 適用於 Ruby V3 的 AWS 開發套件

Warning

有關異步文本分析過程中發生的問題的警告 (StartDocumentAnalysis）或異步文檔文本檢測（ StartDocumentTextDetection。

內容

ErrorCode

警告的錯誤碼。

類型：字串

：必要 否

Pages

警告應用到的頁面的列表。

類型：整數數

有效範圍：最小值為 0。

：必要 否

另請參閱

如需在語言特定的 AWS 開發套件之一中使用此 API 的詳細資訊，請參閱下列說明：

- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- AWS適用於 Java 的開發套件第 2 版
- 適用於 Ruby V3 的 AWS 開發套件

亞馬遜文字中的硬性限制

以下是無法變更的 Amazon Textract 中的硬性限制的清單。如需可變更位置和限制的限制的詳細資訊，請參Amazon Textract 檔案端點和配額。如需有關可變更之限制的資訊，請參閱 AWS Service Limits。若要變更限制，請參閱建立案例。

Amazon Textract

限制	描述
接受的檔案格式	操作支持 JPEG、PNG、PDF 和 TIFF 文件。（支持 PDF 中的 JPEG 2000 編碼圖像）..
文件大小和頁數限制	對於同步操作，JPEG、PNG、PDF 和 TIFF 文件的大小限制為 10MB。PDF 和 TIFF 文件也限製為 1 頁。對於異步操作，JPEG 和 PNG 文件的大小限制為 10MB。PDF 和 TIFF 文件的限製為 500MB。PDF 和 TIFF 文件的頁面限製為 3,000 頁。
PDF 特定限制	最大高度和寬度為 40 英寸和 2880 點。PDF 無法受密碼保護。PDF 可以包含 JPEG 2000 格式化的圖像。
文檔旋轉和圖像大小	Amazon Textract 支持所有平面內文檔旋轉，例如 45 度平面內旋轉。 Amazon Textract 支持所有側面分辨率小於或等於 10000 像素的圖片。
文字對齊	文本可以在文檔內水平對齊。Amazon Textract 不支持文檔內的垂直文本對齊方式。
語言	Amazon Textract 支援英文、法文、德文、義大利文、葡萄牙文和西班牙文的文字檢測。Amazon Textract 不會返回其輸出中檢測到的語言。
字元大小	要檢測到的文本的最小高度為 15 像素。在 150 DPI 時，這將與 8 點字體相同。
字元類型	Amazon Textract 支持手寫和印刷字符識別。

限制	描述
字元	Amazon Textract 字檢測到以下字元： • a-z • A-Z • 0-9 • ä Ä ö Ö ü Ü ç Ç é É â Â ê Ê î Î ô Ô û Û à À è È ù Ù ë Ë ï Ï ü Ü á Á é É í Í ó Ó ú Ú ü Ü ñ Ñ ì Ì ò Ò ã Ã õ Õ • ! 「# $% '& () * +,-./:; =? @ [\] ^ _ `{|} ~ > <° € £ Ø Ø ©®™ § ¹ ²
分析 Zeid 特定限制	AnalyzeID 僅支持美國護照和美國駕駛執照。

Amazon Textract 的文件歷史記錄

下表說明每個版本的Amazon Textract 開發人員指南。如需有關此文件更新的通知，您可以訂閱 RSS 摘要。

- 文件最近更新時間：2019 年 5 月 29 日

update-history-change	update-history-description	update-history-date
集成代碼示例AWS文檔 SDK 代碼示例 GitHub 回購	Amazon Textract 指南現在包含其他代碼示例。將前面的示例部分重命名為教程。	2022 年 1 月 30 日
已新增資訊費	Amazon Textract 現在支持使用分析費用 API 對發票和收據單據進行分析。此功能僅適用於我們的亞太地區（孟買）、亞太地區（首爾）、亞太地區（新加坡）、亞太地區（悉尼）、加拿大（中部）、歐洲（愛爾蘭）、歐洲（倫敦）、美國東部（俄亥俄）美國西部（俄亥俄）和美國西部（俄勒岡）地區。	2021 年 7 月 26 日
Augmented AI Support	Amazon Textract 現在支援 Amazon Augmented AI，以實施人工審核。	2019 年 12 月 3 日
新的服務與指南	Amazon Textract 現在全面供應。	2019 年 5 月 29 日
Support 選擇元素	Amazon Textract 現在可以檢測選擇元素（單選按鈕和複選框）。	2019 年 4 月 24 日

Amazon Textract 的版本	這是 Amazon Textract 文件的第一個版本。	2018 年 11 月 28 日

AWS 詞彙表

如需最新 AWS 術語的清單，請參閱 AWS 一般參考中的 AWS 詞彙表。

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。